

markup

1 9 9 0 e 1 4

/ EXCLUSIVE PREVIEW \

SIXTY FIVE MILLION AND ONE BC

/ THE MAKING OF \

FALLING TROY

/ 2D ARRAY STRING EXPLOSION \

+ VISCERALITY

+ INTERVIEWS WITH

GARETH TILT

2DCUBE

AND

SUPERCASEY4

WELCOME!

Free Games in a Commercial World

Oh, sorry, I didn't notice you there! Let me close down SimCity 4 (the last real SimCity! Don't get me started on Societies). One of the thoughts on my mind lately is if commercial games, just because you have to pay for them, get more attention.

For example, if two YoYo Games staff pick games receive the exact same rating, would one that costs money versus one that is freeware receive more attention simply because of the attached cost? Let's take [Aces High Over Verlor Island](#) and [Sixty Five Million And One BC](#) as the examples. Without thinking about anything else, would you assume 65m+1 was the better game because it was a commercial game?

It's food for thought. And now that there are more opportunities for game maker to sell their games on sites like [pcgamesnow.net](#) we'll probably be seeing a lot more of these discussions pop up. Which also reminds me of those 300 'Games for Windows' CDs that used to be so popular; let's all make mental note not to do that with game maker games. OK? OK!

Now, I've got to get back to my city, the sims need me!

Until next time,

Robin Monks

Contributors

Eyas Sharaiha	Sr. Editor
Robin Monks	Editor
Alaric Holberton	Contributor
Bart Teunis	Writer
Erthgy	Writer
Sam Whited	Writer
Shadow Master	Writer
Suhaib Al-Dari	Graphics Design



Table of Contents

Table of Contents

Exclusive Previews

Sixty Five Million and One BC	4
-------------------------------------	---

Editorials

Going Commercial with 65M+1 BC (Interview) 10	
Viscerality	11

Monthly Specials

Script of the Month.....	18
Extension of the Month.....	20

Tutorials

Making Lists with Local Arrays.....	21
Expanding GM with .NET	25
2D Array String Explosion	31

The Making of...

The Making of Falling Troy.....	33
---------------------------------	----

Reviews

Lab 14	42
Karoshi 2.0	45
A Dragon's Tale.....	49
Ancient Ants Adventure.....	51
MarDar	53

Interviews

Interview with SuperCasey4.....	44
Interview with 2DCube.....	47

news BULLETin

YoYo Games Competition Updates

The results for the YoYo Games Ancient Civilization competitions are out, with [Ancient Ants Adventure](#) in first place by [RedSystem](#), followed by [RhysAndrews's Caveman Craig](#) in second place, and [KC LC's Tut's Test](#) in third place. The 1000\$ Grand Prize winner has been reviewed in this issue, and you will be able to see the reviews for Caveman Craig and Tut's Test in issue 15 of the magazine.

In other news, the third YoYo Games competition theme has been announced: co-operation. The contest is currently open for submissions and the deadline for submissions is August 24th, 2008. You can find more information about the competition [here](#).

C++ Runner and Other YoYo News...

What seemed like the [opportunity](#) to have an exciting wave of news and updates from YoYo Games turned out to be a false alarm. The C++ runner is apparently in the late stages of development and "will definitely be ready this June"; it's August. The YYG Developers also ran into problems with the Pascal platform used in developing the Mac version, which will cause it to be delayed.

I don't particularly have a problem with delays or missed 'public testing' dates really, and I don't think most of the community should. In terms of disappointments, YoYo Games hasn't exactly disappointed us, so we shouldn't really complain about missed dates for the sake of complaining. Still, I hope YoYo's strategy will change; to favor silence over whatever they are still unsure of, *just* for the sake of professionalism.

Panic: Instant Play Breaks on Firefox 3

Panic is no overstatement – YoYo Games' Instant Play feature was incompatible with the latest version of Mozilla's Firefox web browser. YoYo Games promised a fix to the incompatibility, but the fix has been delayed and delayed, prompting members to come up with their [own solutions](#) that involve disabling security features of Firefox 3 to do so, as well as engage in a heavy critique of YoYo Games. An [open letter](#) to YoYo Games was written by Dr. Watz0n of the GMC. Eventually though, YoYo Games did release an update on June 15th that fixed the issue.

While I certainly believe YoYo Games did hurt themselves with such delays to the Firefox 3 update, I still feel the latest unfolding in this is just too much. What do you think?

Third GMking.org 'Audcast' Released

The Third GMking.org Audcast, also known as the Game Maker Podcast, has been released [here](#). In the Audcast, Robin and Eyas discuss various developments in the GMC.

Russell's Quarterly and others...

Tom Russell's game development magazine focusing primarily on editorials on game design has released its [third issue](#). And a new competitor is Game Maker Magazine, whose fourth issue you can check out [here](#). Our friends at GMTech have – like us – halted development somewhere during June but should be ready to have their thirteenth issue on their [website](#) soon.


SIXTY FIVE MILLION AND ONE BC

By Eyas Sharaiha

Back in [November](#), I was certainly lucky to have laid an eye on the latest version of Sixty Five Million and One BC. Today, I'm thrilled to show you an *exclusive* preview of the full version of this game... *only* in MarkUp Magazine.

Overview

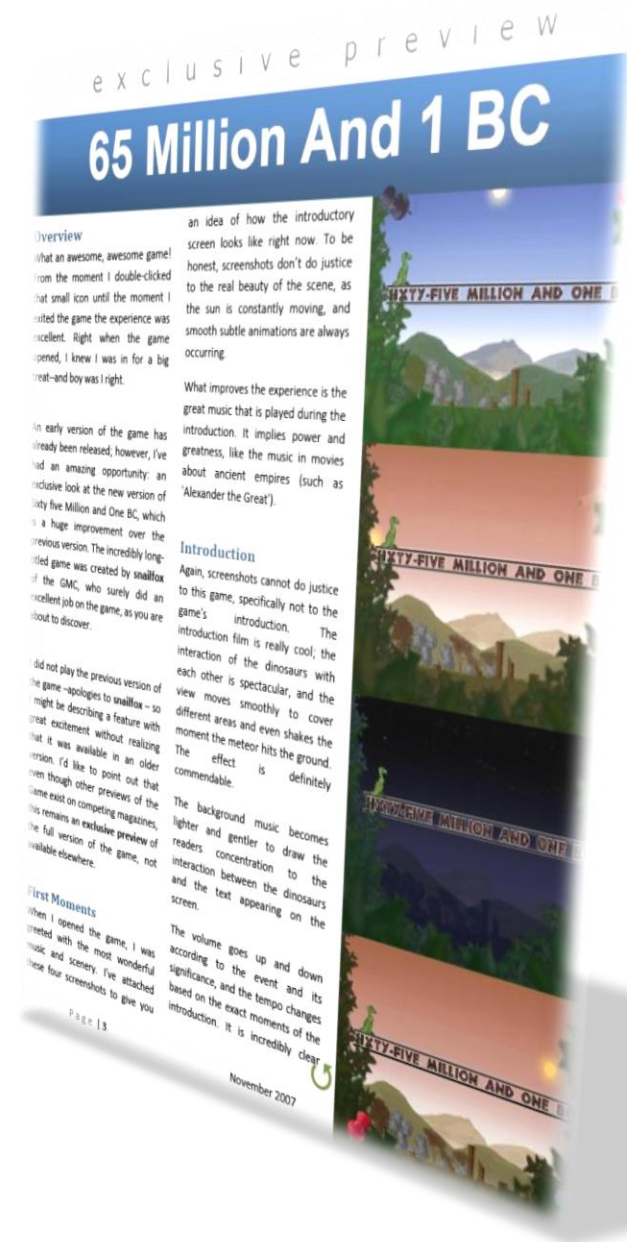
It is generally believed that the reason for the mass extinction of Dinosaurs is the collision of a huge asteroid with the earth, effectively ending the Mesozoic era. In Sixty Five Million and One BC, you travel as a velociraptor around various locations to try and activate and assemble a dinosaur-designed machine that will hopefully save the world. In general, the game is intelligently designed, extremely addictive, unbelievably versatile, and never stops being engaging and action-packed!

 View the Exclusive 65M+1 [Screenshot Gallery](#), [Here!](#)

Starting with the Game

From the moment you double click the game, you are greeted with the *best*-quality scenery, smooth effects, genius music, and subtle sound effects to complement it all.

When you open the game, you are still presented with the 'animated' menu that blew me away in the first preview. Now, the menu options are all available and properly laid out, and menu items can be selected using the mouse by highlighting the options or via the keyboard's arrow keys. Both methods result in subtle click effects that appear when an item is being highlighted or selected.



Exclusive Preview

SIXTY FIVE MILLION AND ONE BC

By Eyas Sharaiha



The game's preferences have a difficulty slider, which I thought was an excellent addition, which makes the game highly customizable. As well as options to remove the music (no idea who would want to do that) or any of the 'supplementary effects' for those on lower-end machines; which makes the game highly optimize-able, I like it!



The game starts rather interestingly with the game's HUD following another character, which you – the main player – finally borrow from the character to begin your adventure across the Mesozoic world. Such light hearted interaction continues throughout the game and will be discussed further later in the preview.



Game Features

HUD

The whole Heads-up display of the game (or the user interface) is extremely smart, but most importantly helpful and easy to use. The **context-sensitive buttons** we have on the top in each window tells us the controls that we are allowed to use at any instant in time, the buttons are usually only the Control button and the Space bar, but their functions and effects change according to what you're doing at the moment. If you're jumping then 'Ctrl' makes you climb, if you're standing still then it makes you tail whip, and if you're walking it makes you run. When a button is pressed its graphic changes to indicate that by highlighting it, and if releasing the button does anything, then the button is properly labeled accordingly.

The **health bar** and **health** in general in the game are also

SIXTY FIVE million and one BE

By Eyas Sharaiha

Context-sensitive Buttons



Health Bar

Star Count

well made. The health bar has this jagged outline that fits the entire atmosphere of the game and correctly reflects the era in which the game takes place. When it comes to health in terms of quantity, I think it has been well calculated to take into consideration the difficulty curve of the game, well done. Also, the screen starts “glowing red” when you are damaged past a certain point, which is cool. Death also results in a quick yet smooth red ‘flash’ on the screen. Overall, the health is properly reflected in the game’s interface, both quantitatively with the aid of the health bar and reflects the ‘mood’ that is usually accompanied by the health and status through the flashing in the interface, the colors, etc..



Next to the health bar, we have the **star count**, which is

the number of stars you have collected. Every 30 stars you collect, you get revived!

Versatile Gameplay

The gameplay of the game is incredibly versatile to the extent that versatility is no longer just a plus or even an asset to the gameplay, but is virtually an indivisible *feature* and part of the game, that makes it the great game that it is.



At times we’re playing a simple platform on ground fighting other monsters. But as you progress through the story, even this simplicity evolves into something with more purpose; collecting components to build a machine and saving life.

At other times, we move to puzzle solving! Indeed, without solving such puzzles one cannot progress in the game, thus forming a crucial aspect of the game play. The gameplay itself changes and shifts every once in a while; it never stops being entertaining!



Other forms of gameplay including racing, frog collection, and avoiding meteors! All these fit properly to the storyline and form a continuous stream of versatile gameplay, never ceasing to surprise and amaze the player.

Sixty Five Million and One BC

By Eyas Sharaiha

Intelligent Design!



The game has been designed rather intelligently, with careful placement of objects, walls, monsters, and even the order of the game and its events.



Whether it's the choice of bosses to fight at any time, or

the way the bosses act, the game will never stop being breathtaking, that's for sure! Each boss needs to be dealt with in different ways, sometimes you need to make them dizzy by jumping around, and at other times needing to reflect their shots, and... you'll even have a chance to make a group fight, where a team of green velociraptors takes on the evil black ones! Whichever boss you're fighting, the game requires massive *skill*, but also incredible *strategy*, and a game that requires both swiftness and intelligence in order to complete it (often both at the same time) deserves every penny, and more.

Other Assets

Music

The demo available on the YoYo Games website shows you the amazing music of the game, but do not think it's just one or two beautiful tracks that repeat over and over again – oh no! – as you progress through the levels, you'll see more and more, all fitting the mood and atmosphere of the game at the time; whether its tense, exciting, light-hearted, or just pure fighting. There are a total of 38 tracks for the game, and not one of them is 'just average'.

Screenshots

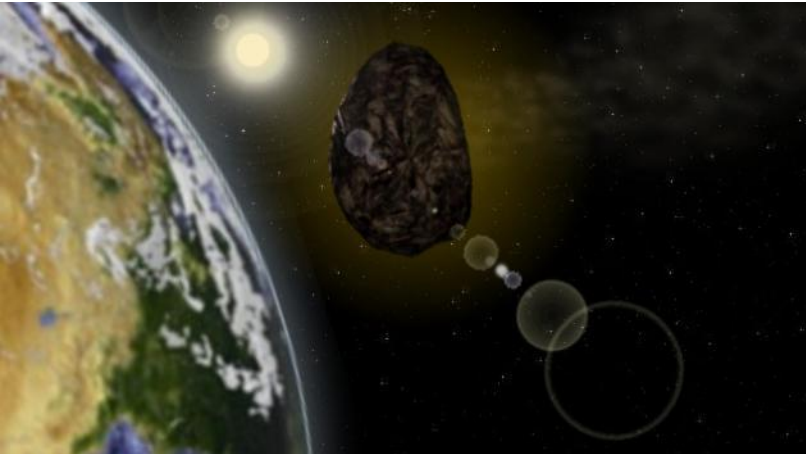


SIXTY FIVE MILLION AND ONE BC

By Eyas Sharaiha

Scenery

The game's amazing scenery and graphics are spectacular and well made. Whether it's in cutscenes or the regular gameplay (although the graphics *do* become generally better in cutscenes) looking at the game screen can be a very rewarding experience!



Graphical Effects

Other sorts of effects exist as well; whether it's drinking, water particles moving, light, fire, the sun, smoke; all of these are well integrated into the game and compliment it very well.



Humor

The sense of humor present in the game is also an asset to the gameplay.



Throughout the game, playful jokes like the ones shown are incorporated. These, as well as spoofs of popular movies and TV shows can be spotted.



The above reference to '24' isn't alone, the game has references and for movies like Deep Impact, Armageddon, and others! The overall sense of humor 'comforts' the player at times, and playing the game might sometimes be just to see the jokes that have been thrown around the game, here and there!

SIXTY FIVE MILLION AND ONE BC

By Eyas Sharaiha

Finishing the Game and Beyond

Inside the machine, it becomes your task to save the world. What happens? I definitely won't give that away, so I tried to take screenshots that don't give away the game's outcome. In general, the game becomes much more intense (both inside the machine and after that), and you're faced with amazing surprises and a brilliant outcome. You'll have to buy the game to see it for yourselves, but I'm telling you: you'll like it!



When finishing the game, you'll have a new button in the menu that stands out: *quick play*.

I'm not sure how you'll feel when you see it, but I was thrilled; I love those cool goodies you get for finishing the game! And this is definitely one of them.

Looking back at my previous preview...

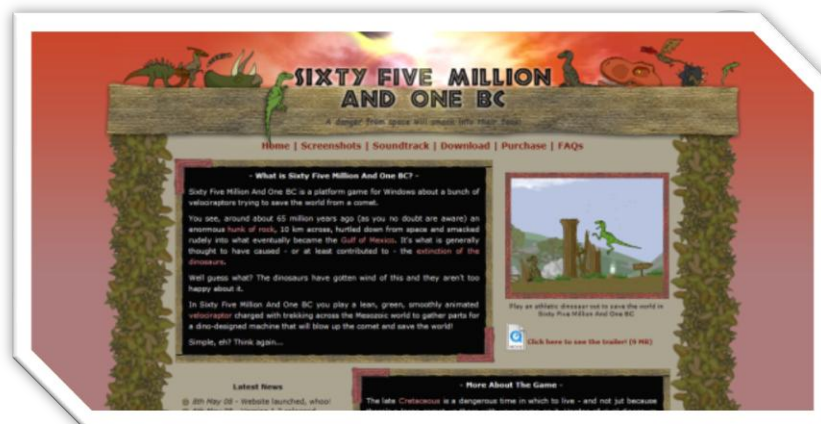
In my Issue 9 preview, I was equally impressed with what I had seen so far, and thank goodness this hasn't changed. Out of all that I had experience throughout the game during that time, I only had two comments: making the difficulty curve smoother, as well as cutting and shortening the extra-long introduction cutscene. I feel both of these have been addressed correctly.

The introductory cutscene was split into two parts, with a HUD introduction first, then learning how to move, then another cutscene to introduce you to the storyline of the game. Well done! As for the difficulty curve, I either became better or the game's difficulty now increases in a more graduate rate such that the player's skill keeps up (most of the time) with the game's increasing difficulty. The difficulty slider is also there to make things better, of course. The game is still very difficult, but that is what makes completing it such a rewarding experience!

Conclusion

The problem I face when writing reviews for games of such quality is that I always run out of words! 'Amazing', 'genius', 'spectacular', 'brilliant', 'intelligent', 'exciting' are words I've used so much already, and some might even be ticked off at the redundancy already! Well what can I say? Definitely one of the best *games* I've ever played, and so far, it is for sure *the* best Game Maker game I've ever seen; it's perfect, complete, and playing it is an experience that goes full circle. You'll miss out on a *lot* if you don't try the demo, and I surely recommend purchasing the game. Excellent.

Visit the Game Website at 65millionand1bc.com. There you'll find all the information you need to try a limited demo or purchase the full version.



Going Commercial with Sixty Five Million and One BC

By Eyas Sharaiha

In their words

Sixty Five Million and One BC is a platform game for Windows about a bunch of velociraptors trying to save the world from a comet; everything past that is details!

From a 'Commercial Game' standpoint, the game is not shareware, as the free downloadable demo is separate from the full version; the demo cannot be unlocked to become the full version, instead, when someone purchases the game, he or she will get a self-contained executable of the full version. This could aid in security and avoiding 'cracking' the game.

Limitations of the Free Version

- Contains only about 25% of the gameplay (and storyline)
- Cannot watch final cutscene and see what happens in the end
- Cannot unlock the game's "quick play" feature to replay your favorite bits



Interview

I had a chance to talk to **Gareth Tilt**, the creator of Sixty Five Million and One BC, and see – very briefly – his opinion about the path he took with his game commercially and his advice for other future commercial game developers.

Why did you use the distribution method you chose?

Using a software registration service meant that I didn't

have to set up my own payment system. It was a practical time-and-effort-saving option.

Did you initially plan for the game to be commercial?

Not at all, I would have been horrified at the idea of charging money for one of my games a couple of years ago. It only started to seem like an attractive idea when the game was nearing completion and generating lots of positive feedback from testers.

Do you believe your distribution method has been successful?

Yes, about as successful as an old-school Game Maker platform adventure game can be in the real world!

Do you have any advice to those planning on making commercial games?

I think it probably helps to spread as much awareness of your game as possible while it is *still under construction*. Doing so helped me a lot to ask for preview articles in a couple of the Game Maker magazines.

Conclusion

As an indie game developer wishing to survive in the commercial game market, it is important to be able to realistically define the word 'success'. In my own, self-proclaimed 'correct', definition of success, 65 Million and One BC has certainly been a successful game, both as a demo amongst the YoYo Games and Game Maker communities, and as a commercial games across a base of dedicated, albeit niche, users and fans.

Viscerality

By erthgy

Viscerality (Vice-Er`-Al-it-ee) is a term I stole from the Game Informer magazine by GameStop. Viscerality has to do with how much a player is sucked into the game, how blind he or she is of his or her surrounding(s).

The most common perceived use of Viscerality as seen in modern day professional video games is (in order of Genre from best and most noticed to least and lesser noticed):

1. FPS (TPS and Strategy shooters included... saved my butt from all you Gears of War fans, myself included)
2. MMORPG
3. RPG
4. Sports
5. Platformer
6. Fighter
7. TDS
8. Puzzle and card games
9. Other unmentioned genre(s) of games.

So let us look at the top dog of Viscerality: FPS. I will speak of others, and if you are currently creating a game that is not of the FPS genre, you should try to think of how the same principles might apply to your game's genre.

The categories to come are the important elements of Viscerality, and their successful integration and implementation in games is what makes FPS games, or any other genre for that matter, successful.

Gameplay

If you truly are reading this from some form of a professional game creator's standpoint, gameplay is the absolute, number one meaning of games! Think, where would the video game industry be if it was not fun?!

Another question is, because games are generally fun, how much *will* this industry move further in the future?

Anyhow, if Gameplay is the whole core value of games, then why do some of the games on the Game Maker Community (some on YoYo Games, or even commercial developers) sometimes have horrible gameplay? This is the absolute toughest thing to implement under extreme technical terms (as in, it's tough to define, create etc.) and we should praise anyone able to create a game that is fun, *no matter what* the graphics, sound and music, and/or overall design is!

Obviously I am not trying to make it sound like graphics, sound and music, etc. are not worth it, so I will start a long process of "defining" gameplay.

To start, let's look at one of my personal favorite games in the history of YoYo Games (and Game Maker) so far, and that is: Karoshi. Why did I choose Karoshi of the oh-so-long list of titles of freeware games out there? Simply put, game is fun while not repetitive. It is rewarding according to those that stick through zDcube's puzzles until the end. It has more than one gameplay mode of which stand out in their own unique way with the others, an easily navigate able menu for quick replaying of the levels, the levels are short and aren't too hard to figure out, good graphics, the storyline isn't monotonous, and yet still fun to follow through. The biggest thing that fuels fun – in my opinion – is creativity. A big factor that plays a role into adding creativity into a game is originality, proper documentation, planning, and (most importantly) happiness for the developer. If you plan for your game in an organized way, then **while you are developing** your

Viscerality

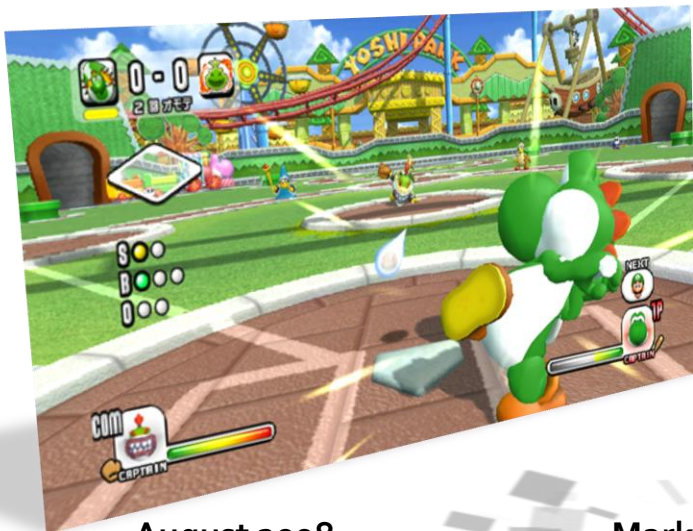
By erthgy

game, you will notice tremendously fun ideas implemented into your game. I find that the best time for me to plan is right before I go to bed, as that is when my mind is most active (as well as a large percentage of people). Another large trend you will need to unveil your skills and time in creating your game on is feedback. This is almost 70% of revising and checking your game (maybe more!) So it's extremely important to do so, even if the cost is another week of seemingly "monotonous" development.

Just make sure your game is fun *before* you release it, simple as that.

Good Graphics

Not just good, but fitting as well. The idea of this is to get the players to stop looking at the real world outside of the monitor or TV box, and to get them to start gazing at your extensive world in-game. The fact that it is that 3D makes you lean forward a little bit or squint your eyes, because the graphics are optically illusioned to look further away, and to keep you on the edge of your couch, waiting for the next graphical display to appear and 'come closer'. While in 2D games you would be sitting in your seat normally, still concentrated on the game, hardcore 3D gamers, will



August 2008

have a little trouble adjusting to 2D gaming, and vice-versa. A hard thing in creating graphics is proper shading. Make sure you look at as many shading techniques, and **change them in game** as it progresses to make it realistic.

Example: Gears of War, New versions of Mario, etc, (you barely have to look).

Expert Sound and Music

Music

This could arguably be the number 1 reason of Viscerality, but, for one thing, it is hard to compare/contrast sound waves and pixels, secondly, it is another for one to compare/contrast different media forms, and let alone game genre's! Anyhow, fitting music not only invigorates the body and the soul, but also influences the player to enact greater acts of valor in the battle field (or against the enemy AI or whatever you want to call the challenge in your game).

Example: In the classic strategy hit title of 2004, *Rome Total War* the music changes according to what happens in the battle field. The most noticeable one is when your legions get attacked from your flank with a cavalry (not Calvary, this is often mistaken by many Americans, myself included since I almost wrote it) charge. Once your unit gets hit, the music (drums, guitar, and vocals) switches to a fast paced beat. Activision and Total War created this in such a unique and almost perfectly timed way that it makes the player want to counteract this attack, twisting and mixing the music in with the gameplay.

Sound Effects

Sound effects are sort of the feel of a game, a gloomy sound should happen when you fail to complete your objective, and a happy sound should happen when you

MarkUp Magazine

12

Viscerality

By erthgy

defeat a level. Many games at the GMC lack this very important quality in gaming. The best ways to create your sound best fitting (some would call it higher quality, but like I said in graphics, the 2 are important but very, very different) is to think about how the object functions, the graphics of objects in your game, what sound would this action make if this happened in real life, and when to put the sound in (as in, which event). Sometimes professional video game developers have even tried sounds in real life for physics sake, just to get it right. I suggest (if possible) that you try to re-enact your game's sound effects in real life. Just to get the physics right, and especially in platformers as they involve a lot of physics

Example: Many of you are creating a game for YoYo Games' third competition. So let's say that your game is a puzzle game that is based off of a civilization. You have to move blocks to a certain form for them all to be deleted and complete the level. These blocks are heavy, brown, and move slowly when you interact with them. When the player moves these blocks, you should make a slow, deep sound since the blocks are heavy, and the heavier the rock, the deeper the tone of sound.

Room Design

A lot of Mario type games have Room Design implemented in them incredibly well. The fact that "Ooh, they put that there, and wow, I found a secret here..." It all adds up to a very well put together room (which in turn, gives better gameplay, leading to an overall better game). A lot of people, (when they are rating a game, of course) combine Room Design with Gameplay layout, as Room Design is arguably half of Gameplay. I do not necessarily have a problem with that, as long as they realize sooner or later, (when they are creating their games) that Room Design is a stupendously awesome power that is essential to all games, and should not be underestimated.

Example: A few examples are Gears of War; the company (Epic) who created the game knew EXACTLY how to put this cover here and face it there, precisely so that it creates the power and weakness that had been necessary for the game to be balanced. The whole idea behind Gears of War and getting used to its cover system is to find the perfect combination of hiding until the last possible second, and shoot, then chainsaw or melee out (according to your weapon) the enemy.

Another commonly known example is any Mario game for just about any console (Wii included, I'm not that much of a die-hard 360 fan!). You have to wait for the perfect opportunity to get from one part of the level, to the next part (normally left side of one level, to right side of the same level). Especially on the later levels where players find themselves waiting for the perfect opportunity to dodge the moving walls, get to a safe spot where a wall cannot squish you, and wait for the walls to move and create spaces for the player to platform on. Then you repeat the process, trying as hard as you can to get as much of the coins as possible to beat your high score.



Comfortable Controls

You really do not need to worry too much about controls in your game unless if the game is "a genre of its own kind". If your game's controls are unfitting; then your game audience (meaning pretty much anyone in the case of the GMC) will presumably stop playing the game after 2-4 minutes of gameplay. Think, if someone is describing a shooter to you, you would say "Oh and I guess the controls are WASD". Think how awesome it would be if the person they were talking to responded like "Actually, the game has the idealized form of WASD into it, but it is so advanced and strategically remarkable that you don't fairly notice the control sketch."

The biggest (non-FPS) problem and example I can think here is the TDS genre on the Game Maker Creation's section of the GMC: About $\frac{1}{3}$ to $\frac{1}{2}$ of the games created there force you to use arrow keys instead of WASD. This is very hard on hardcore FPS fans, as they are not used to such a close squeeze of your mouse and keyboard hand(s); in fact, none of the 3 types of gamer audiences actually feel this style of controls is comfortable. Besides, it's not that hard at all to implement such controls, and to tell you the truth. (*Cough*right click on event arrow keys for main player and left click on the appropriate WASD key*Cough*. Man, my cough from old man winter still hasn't left me!) Just make sure the controls are unnoticed and comfortable when your targeted audience plays your game, and they will be of good quality. Another thing that really will help you is to ask the testers of your game as to whether or not they feel as if the controls sketch was appropriate or not, they give back feedback, which is very likeable by any developer.

Replay Value

Replay Value not only consists of having a game that is fun

to play more than once, but also of creating games that are not repetitive in any way, shape, form, ability, function, etc. Many times have I played games that are on the Game Maker Community, (and also the YoYo Games Community) have extreme resistibility to the player. We have to go back and test our games before we release them... Not just us, but at the very least one video game fan (other than you) of your targeted audience, and at the very least one video game fan not in your targeted audience to give you feed back of your game. After they have played through the game once, have them play through it another time, ask them how well they thought the game was made, then finally have them tell you what they would have different (or they might tell you that your game is just "plain awesome"). The only problem with this is, many people grade games inaccurately, other people are just plain bad at explaining things, or unwilling to actually grade the game a fair treatment and will unfairly be like "good" or "not good" with no explanation of a "why" whatsoever. So make sure that you get a good form of judges when you are looking for Quality Assurers. This ensures that your game is fun or shows you what you need to improve on. (Back to resistivity), play through your game as if you were completely new to the game and/or game genre, but you are an average video game player, pretend you play video games maybe less than an hour to three hours per day, are the game's controls too hard to pickup if you haven't played the game or any genre like it before?

Now here are the biggest/common mistakes I notice on a game that are too repetitive are:

Not enough save files per game

This is quite self-explanatory; quite frankly though, many developers miss this because they test their game out so much that they are too good at it. (Therefore making it quite unnoticeable, as they don't need to use it as often,

Viscerality

By erthgy

and discover of its deficiencies/annoyances as often as they should.) If your levels are long or even moderately sized, hard, or even repetitive, then you should *at the very least* put a save file in the middle of such levels. Perhaps another way you could do this is to have a hidden semi-secret item, that – when obtained by the player – could save the game and give the player points (not to mention self-confidence).

Not having enough challenge

Generally speaking, if your game doesn't have enough challenge, you either did not program enough AI into it, need to do some form of room re-designing, or add more gameplay elements into your game (that sort of goes along with AI, but I just decided to put that in as an add-on because it sounds cool). Another way to add more challenge is by putting hidden items in your game. The player must fight or get past so-many legions of enemies to obtain such item (and earn Achievements if we are talking about the Xbox 360, your choice though!). Just make sure that the game has more than one way to defeat it if you decide to add a hidden item system, so that you can still complete the game without such secrets, and keeping that feeling of accomplishment to finding such items. Think, will your game's secrets alter the game's outcome? Or will they merely add a point or two to the high score list? Are they too hard to find? Again, ask the people who test your game.

Not having enough lives

This goes along with "Not enough save files per game", but in games like Mario, it is extremely essential that there are enough lives to vary it up a bit (which gives the user select mobility, which is very good). If there are not enough lives in the game, and the player dies many times per level... Then I am sorry to say this, but, your game will be too delicate to play for most audiences that might play

the game to begin with, and therefore players (even good fans) that actually try to beat the game, will probably get driven away before beating it.

Bad HUD

HUD stands for Head-Up Display. Some people don't know what a "Head-Up Display" is, but a HUD is really the player's mini-map in a FPS, score counter in a platformer, and other things besides the games' screen that are trackers or useful add-ons/controls to the player. It is just *downright annoying* for games to have bad HUD's. Yes, I'm talking about the ones where the screen is only $\frac{1}{4}$ of its normal size because of an oversized, unkempt HUD. If your game does have a bad HUD, then EVERY SINGLE TIME players play the game, they will seem to be pushed away from your game. Many games don't have this problem, but it can be overlooked on the developer's part.

Bad Scoring system

Now, I'm not saying you need a scoring system like Microsoft's Xbox Achievement awards, but your game's scores need to have a *varying output*. In reality, the only way to have a *varied* score system is to have high numbers and ranges. Sometimes, you will have to use math and creative ability to get points in a platformer like Mario. Hey, have you ever wondered why Mario's points are so numerous? It's because Nintendo wants variation in that particular system, they want some things to be different than others. Think of how much of a cheap idea it would be, if players had to practically beat the game 5 times, looking for secrets that are only 1 point. While at the same time, there are other objects in plain view that are worth the same value. A good way to balance your game (point-wise) is to use math. It really isn't that hard either; you just have to think... For instance, in a TDS where a player might get **10** points for killing a zombie, say there's a boss that takes **60** times as many normal shots to kill, how

Viscerality

By erthgy

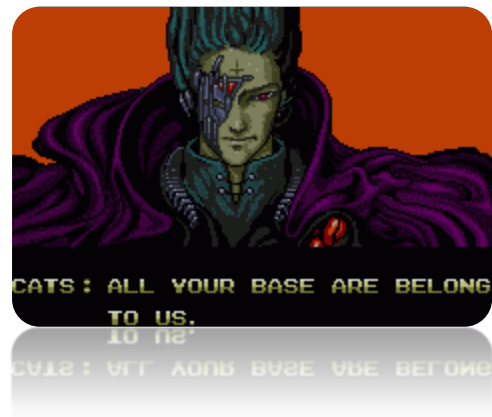
many points should the boss reward the player for defeating it? Well, if the original zombie took 1 shot to kill, and the boss took 60 shots to kill, the 1 times 60 is the mini “formula” we will use for this: so, 10 points x 60 points = 600 points. There you have it; we magically balanced your game’s scoring. Other times, it is good to sort of “unbalance” the game if this “formula” type system disrupts the game’s creative flow, it is fine to just give it its own set of point value. The tough part here is determining whether or not 2 things are related, though.

Example: So, let us say that there’s a different enemy that doesn’t get killed from shots, maybe just an energy gun (perhaps one like in Halo) and the energy from the gun recharges after a certain time, according to how powerful it is. This is the case where you would want to create a different health deduction formula, you would probably base this health system on how long the recharging of energy takes, how the gun shoots, how hard it is to aim, how much power it has, and finally, how fast each shot is. Just do whatever fits your game the most, but remember, always highly consider of the variables put before you, as an unbalanced health/score system in game shows unprofessionalism (or in our case, De-Viscerality!).

Bad English

Sometimes I do feel that the problem of bad English is continuously ignored by game developers. They think “Oh, well, English doesn’t matter”. Another thing they may think is, “Oh, well if it sounds right, I’ll use it”. There are quite a few things wrong with that quote right there. Firstly, if you don’t care about the English rules in the first place, then how the heck are you supposed to know if something ‘sound’s right’?! Secondly, what if you heard it incorrectly, or “speed-read” through a text box’s text (which you thought you were correcting) in your game? **Humans cannot rely on themselves to detect and neutralize their own problems in life.** There, now that I got that out of my mouth, let’s get into the actual English

teaching. English is not necessary for life. There, I said it. (Now I’m feeling the burn), but remember, English *is* necessary to create a good impression for doing business or trade with someone else (most likely, other businessmen, (or businesswomen)). Think, how unprofessional would it look if in every game *ever* created, the letter “I” had been un-capitalized; there had been no punctuation marks, incorrect spacing, wrong spelling, word infusion (i.e.: Making up words like “shisn’t” or something weird like that.) Any person that correctly knows, (and cares!) for the sake of proper English, will correct you. I will give you a list of common English mistakes, (I’ll list the incorrect word/phrase, then the rule, then the correct word/phrase. Also note that these are not in any order, but please do check them before distributing your game, or even better, before you start scripting (plot wise, like writing the story line or message box in your game, not GML code!)) To help you people out, I put the incorrect version of the English words in italicizes, and quotes.



Are and is

“You students is in big trouble!!” Why is this wrong? Because the rule states “A singular noun has a singular verb, and a plural noun has a plural verb.” (Plural simply means **two or more**, singular means one, this has to do with the number of things, in our case “students” would be plural, as the s at the end of it indicates **two or more**. Ok, so now what would the corrected

Viscerality

By erthgy

version of that phrase be? "You students **are** in big trouble!!" See the difference? Since "students" had an s at the end of it, that would make it plural, and "are" is the plural form of "is". (Remember: Another way to remember "if a noun is plural or not" is to see, in the context of the word, is it talking about one? (If so, that is singular, notice "**single**" is the base, (or first) part of the word "**singular**".) OR, is it talking about *two or more*? In our case, it would be plural.)

May and Can

Oh my goodness! May and can are the absolute most hated words to any English scholar (or anyone smart in English). "*Can I get a hot dog?*" Here is the rule for May and Can "May is used to ask permission, can, (on the other hand) is used to ask "*is it possible to do this?*" So, **technically**, that question in the quote which I said was wrong, isn't wrong. The only problem is that... If you needed to ask *if it was possible* for you to get a hot dog, and all ready knew that you had permission to do so, then why not just get up and get one? That is why (we assume) that the quote is wrong, because the person was asking **if he could have permission**. So, the correct way to ask the phrase is, "**May** I get a hot dog". See how much smarter you're sounding all ready?

Capitalizing "I"

"Once i looked over this rule, it was very evident to me that we (humans) are very selfish to make up such a rule as this (you'll understand further as I elaborate (or explain) this rule." The way the rule works is if you are writing English then the word "I" is ALWAYS capitalized. Additionally, make sure "I" is always capitalized, so if you also meant something like "The letter *I* is my favorite letter", then you would still capitalize it, but put it in quotes if you are hand writing, or italics if you are on a PC.

Than/then

"*We went to the store, than went to the video game shop.*" Why is this wrong? For this particular example, we will have to look at the meaning (sometimes called context) of each of those above words. 'Then' is used as a continuing word in lists that connect

each other together. One way to help define "then" is to replace it with the phrase "next in line" and see if it makes sense. Ex: We went to the store, **then** to the video game shop.

As for the word "than" could be replaced with "instead of" and see if that makes sense. For those that are all ready English smarty pants, notice that "than" could also be considered as a coordinating conjunction, meaning that it connects (in a way) two independent clauses. (If those sentences up there confused you, ignore them.) Ok, here is our example of the use of the word 'than': "I would rather you buy Halo 3 than Super Mario Galaxy, just because you don't own a Wii." Now, try substituting the word "than" in those quotes with "rather than", and notice, it works.

Conclusion

Thank you for reading my long discussion on Viscerality, by the next issue (or two) of Markup, I will have another article about this subject with charts, tips, and strategies to make your game the best as it can be. But for now, good luck, and I hope you have enjoyed reading this article (as much as I enjoyed writing it).



Background

The determination of the normal to the point on a surface is incredibly important in game design in many ways; it allows you to achieve more realistic collisions, allows you to achieve smooth collisions with a wall object (where a player continues to move along the wall after colliding with it at a certain angle), and much more. Therefore, I present to you the `normal_detect` script, which returns the direction of the normal to the surface at a given point.

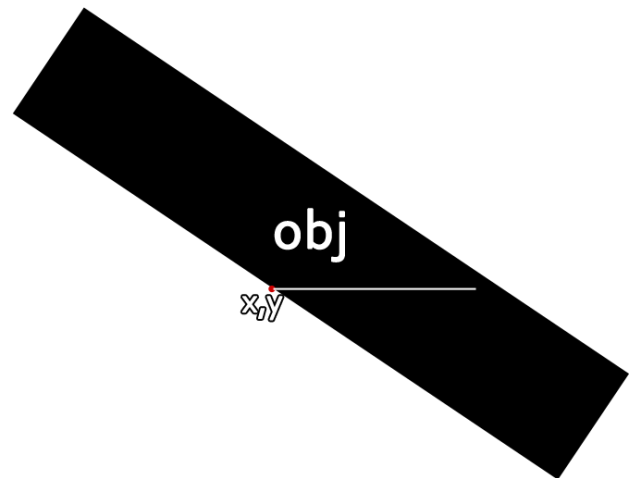
The Script

```
/*
** Usage:
**   normal_detect(x,y,obj [,rad [,res]])
**
** Arguments:
**   x,y   point on the surface
**   obj   an instance or object
**   rad   radius of test area, positive int, optional (default 4)
**   res   resolution of test, positive int, optional (default 1)
**
** Returns:
**   a surface normal (in degrees), at the given point (x,y)
**   on or near the given (object) by scanning an area of the
**   given (radius) and (resolution), or (-1) on error
**
** Notes:
**   Makes approximately pi*radius*radius/(res*res) collision calls.
**
** GMLscripts.com
*/
{
    var xx,yy,obj,rad,nx,ny,i,j;
    xx = argument0;
    yy = argument1;
    obj = argument2;
    rad = argument3;
    res = argument4;
    if (rad <= 0) rad = 4;
    if (res <= 0) res = 1;
    rad += 0.5;
    nx = 0;
    ny = 0;
    if (collision_circle(xx,yy,rad,obj,true,true)) {
        for (j=res; j<=rad; j+=res) {
            for (i=0; i<rad; i+=res) {
                if (point_distance(0,0,i,j) <= rad) {
```

```
                    if (!collision_point(xx+i,yy+j,obj,true,true)) {
                        nx += i; ny += j; }
                    if (!collision_point(xx+j,yy-i,obj,true,true)) {
                        nx += j; ny -= i; }
                    if (!collision_point(xx-i,yy-j,obj,true,true)) {
                        nx -= i; ny -= j; }
                    if (!collision_point(xx-j,yy+i,obj,true,true)) {
                        nx -= j; ny += i; }
                    }}}
                if (nx == 0 && ny == 0) return (-1);
                return point_direction(0,0,nx,ny);
            }else{
                return (-1);
            }
        }}
```

Explanation

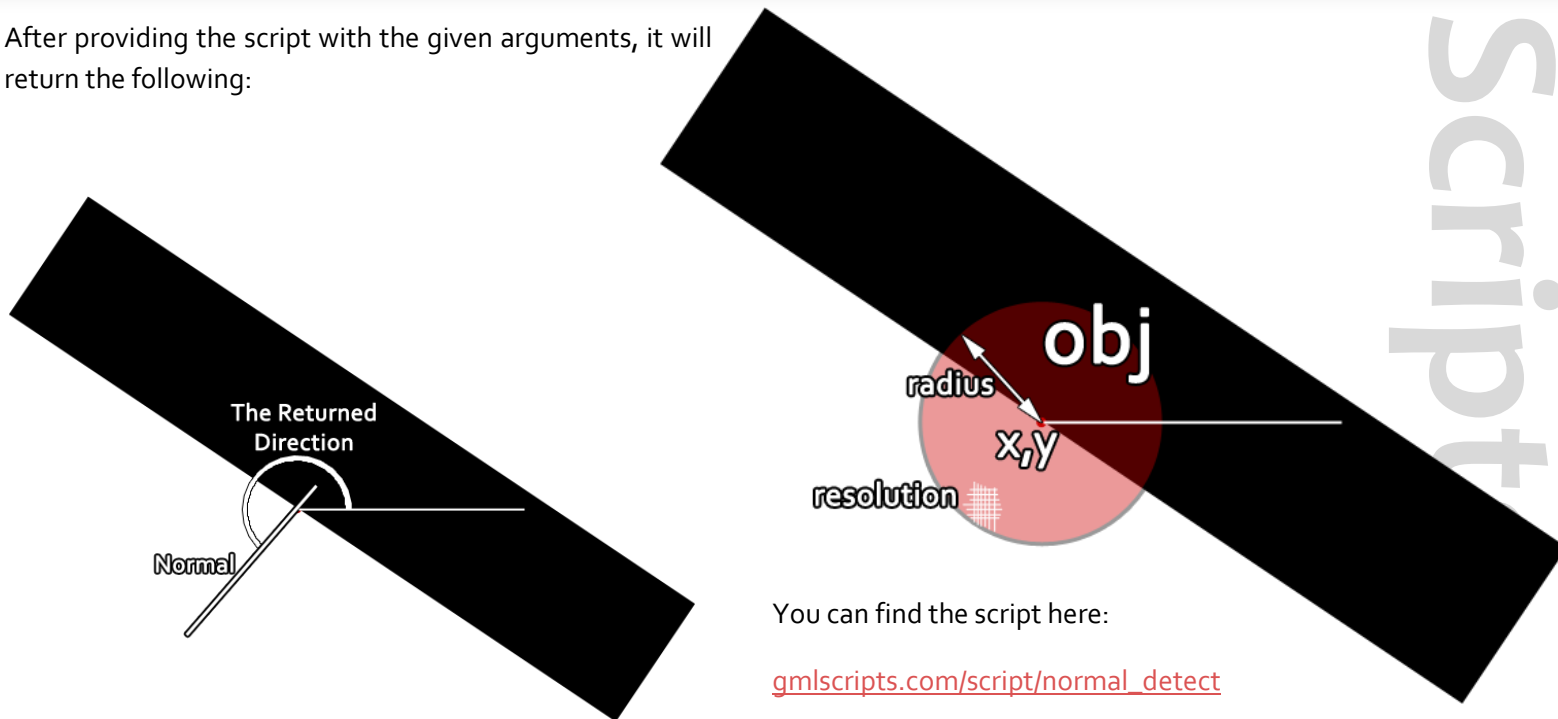
The basic required input to the script is simple: two points, x and y that refer to a certain position on the surface of an object, and the name of the object or the instance of an object which is being referred to.



SCRIPT OF the month

Powered by GMLscripts.com

After providing the script with the given arguments, it will return the following:



You can find the script here:

gmlscripts.com/script/normal_detect

The returned direction is of course a typical Game Maker direction value: it is in degrees and measured counter clockwise beginning from the right.

Alternatively, you can supply additional arguments to tweak the script to your taste.

Other than the object and the x and y points on the surface, you can also check the radius of the area to be scanned. The default radius is 4, but if –for instance – you need a more accurate value, you might consider increasing the radius for instance, however if for instance the radius superseded the width of the object, you might get wrong returned values, etc. The resolution is the amount of space between every two checked points; it is 1 pixel by default.

The Script

Special thanks to xot for creating the script.

Conclusion

This is a pretty awesome script I must say, and I'm glad I found it! This has a lot of potential benefits in Physics, motion, and interaction with objects; I hope you make good use of it!

extension of the month

Powered by GMBase

The INI DLL

Kyle_Solo created a new extension: the INI DLL. To quote the (very quick) description of the extension:

INI DLL lets you:

1. *Edit multiple INI files at once*
2. *Edit INI files in any folder*
3. *Does it faster than any other system (including Game Maker!)*

How this is better than Game Maker

One of Game Maker's limitations when it comes to handling the INI file format (which is an excellent previous- and current-generation data storage format) is that it only reads INI files in the current location, and this extension overcomes this limitation.

Another limitation of Game Maker is that it is only capable of reading INI files; while file writing functions can be used to edit the INI files, the process is rather manual, keys and sections cannot be easily found and modified, etc.

This extension, on the other hand, is capable of deleting keys of your choice as well as entire sections, checks for the existence of keys and sections, and reads various data-types from the INI file as well: integers, strings, and doubles (equivalent to 'real's in GM).

The INI Data Storage Format

INI is most frequently used to store game settings, but that doesn't take full advantage of the INI file format; the



concept of sections would only be used in game settings INI files to make the file 'neater' and not more. INI files can store level settings to multiple levels out there, level editing information, stats and different numbers (damage, speed, strength, health, etc.) of the various enemies in your game to allow yourself to have multiple types of enemies from a single object or 'template' by simply changing numbers, values, and probably references to image files or sprites, etc.

Conclusion

Overall, the INI data storage format is very useful and this DLL extension allows you to take more advantage of it.

Get it now:

gmbase.cubedwater.com/?page=extension&id=177

making Lists with LOCAL ARRAYS

By Bart Teunis

In Game Maker version 6.0, functionality to add data structures was added. Data structures offer an easy and efficient way to store and manage data. They are only available in the Pro edition of Game Maker, but it's actually pretty easy to implement the same system yourself. This article will explain how to do this for list structures. Other types of data structures can be made in a similar way.

What is a data structure?

A data structure is simply said a collection of variables. Data structures can be compared to arrays, although they're a bit more advanced and easier to work with. They're not only used in GM, but also in other programming languages. GM offers 6 data structures to work with: stacks, queues, lists, maps, priority queues and grids. A stack can be compared to a pile of dishes in a kitchen. The last dish that's put on the pile is the first that will be taken off (it would be difficult to take the dishes lower on the pile). A queue probably doesn't require an explanation. Lists can be compared to single-dimensional arrays in GM. At each index, an array contains a value. The same is true for lists. Maps contain key-value pairs. This is a very handy data structure to make an inventory, for example. A priority queue is the same as a normal queue, but the values have a priority. And finally, grids can be compared to two-dimensional arrays.

So now you have an idea of what data structures are and what they can be used for in GM. Time to have a look at the implementation.

The Basic Idea

Lists can be compared to single-dimensional arrays.

According to the GM manual, they're even implemented using arrays. It is obvious that we will use arrays to store the actual data for these data structures.

Lists have a few advantages compared to arrays:

- They have an id. ⇔ Arrays do not have an id or any reference.
- There is no limit to the number of values in a list. ⇔ Single-dimensional arrays can only contain 32000 values.
- They can be destroyed so that the memory they're using is freed. ⇔ There is no way to destroy an array, unless the object that contains the array is destroyed. Global arrays cannot be destroyed (you can interpret these arrays as being local to the program, so they're destroyed when the program ends).
- GM has functions to manage lists. ⇔ There are no functions to insert or delete a value from an array. This needs to be done by using loops.

Keeping these things in mind, we'll try to find a way to get the same functionality. A first possibility (using a global array) would be the following:

Implementation of lists using a two-dimensional array														
	0	1	2	3	4	5	6	7	8	9	10	11	12	. . .
0	5	a	b	c	d	e								
1	-1	-	-	-	-	-	-	-	-					
2	12	88	64	29	13	4	79	38	56	95	10	21	87	
.	↑													
.	size													

making Lists with Local Arrays

By Bart Teunis

A two-dimensional array can be seen as an array of single-dimensional arrays (lists). That way, a two-dimensional array can be used to store lists. The row number represents the id of the list. The first value in each array is used to store its size. For a list that has been created and also destroyed, -1 is used. This does mean that this memory space can no longer be used since the counter for list ids is only incremented, not decremented... This is not the only problem. Suppose a single list is needed. The first row is the first list and this is the only row that should be filled with values. Now suppose it is filled with 10000 real values. This means that $10000 * 8 \text{ bytes} = 78,125 \text{ kB}$ of memory is required. No problem, you'd say. But GM allocates memory for the array in both dimensions. That means that the actual amount of memory used is $10000 * 10000 * 8 \text{ bytes} = 763 \text{ MB (!)}$ for a single array of 10000 items. That's just ridiculous. It's clear that this possible solution only causes more problems. And it wouldn't work for other data structures, either.

A second possibility uses a completely different approach. GM uses objects. These objects can have local arrays. When an instance of an object is destroyed, the memory used by its local arrays is also freed. And instances of objects also have ids. Now that's nice.

The solution is obvious now. We use an object that serves as a blueprint for list instances. Creating a list actually means creating an instance of the list blueprint. This instance has a local array of values (the actual list) and a variable to keep track of the list's size. When the list is destroyed, the instance is destroyed and the memory used by the local array is freed. And viola, that solves the memory issue. A small disadvantage: GM objects have a lot of built-in variables that we actually don't need. However, the amount of memory used by these variables is negligible. Writing `ds_list` scripts also becomes very easy. The list id (instance id) can be passed as an argument to a script. Internally, the script uses a `'with'` statement

with that id. The dot operator (`.`) can be used as well.

This solution clearly has many advantages:

- easier to implement because objects and instances already have id's, which is exactly what we need
- memory is freed after the instance (and thus the array) is destroyed
- easily usable to create other data structures (queues, stacks, maps, grids, ...) as well
- no longer a single, large, global array to store all arrays

The limit of 32000 items will remain, though. It won't get much better than this. So now that we know all this, let's have a look at the implementation.

Implementation in Game Maker

The first thing we need is an object that will be the list blueprint. We will name this object `obj_list`. As an unregistered user, you cannot use the `object_add` function. That means that you need to add `obj_list` yourself each time. That's not very user-friendly. We'll find a solution for that later in this article.

As explained in the previous part, this object needs a size variable. It is not at all necessary to add this variable through a piece of code in the object editor. This will be done in the list creation function (`ds_list_create`). Time to have a look at it! As mentioned before, creating a list means creating an instance of the list object:

```
var inst; inst = instance_create(0,0,obj_list);
inst.size = 0;
return inst;
```

The script returns the id of the instance. When we speak of lists, this is the id of the list. Note that, for the user of this

making Lists with Local Arrays

By Bart Teunis

script, it works in the exact same way as the original `ds_list_create` function. Also note that here, the "." operator is used to set the size variable for the instance. The following piece of code is just as valid:

```
with (inst) size = 0;
```

Destroying a list means destroying the instance of a list object. The `ds_list_destroy` script looks like this:

```
with (argument0) instance_destroy();
```

A problem that can occur here is that the user of this script accidentally passes the id of an instance of another object. To avoid this problem, the following condition can be added:

```
if (argument0.object_index == obj_list)
```

This condition can be added to all scripts that require a list id as an argument.

Now let's have a look at some other list functions. In all these functions, `argument0` is the list id.

Clearing the list (`ds_list_clear`) is simple, too. The size needs to be set to 0. This doesn't clear the array from memory, but it's the only way to "clear" the list.

```
with (argument0) size = 0;
```

Return the size of the list (`ds_list_size`):

```
return argument0.size;
```

Check whether a list is empty (`ds_list_empty`):

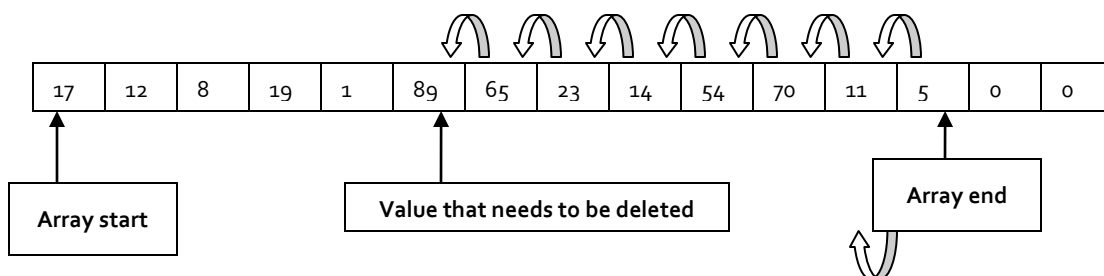
```
return (argument0.size == 0);
```

The `ds_list_add` function is the first function that actually sets an array element. Before that, the instance doesn't even have an array. We'll call the array `array` (yes, it's very unoriginal). The script then looks like this:

```
with (argument0)
{
    array[size] = argument1; //argument1: value to add
    size += 1;
}
```

Can the `size += 1;` statement also be put before `array[size] = argument1;`? It depends. When the size is incremented before the array value is set, array indices will start from 1. When the size is incremented after the array value is set, array indices will start from 0. List indices in GM start from 0 so we will put the increment statement after the assignment.

Deleting a value is a bit harder to do. The simplest way to do this is as follows: when a value is deleted, all values with a higher index than the value that needs to be deleted move 1 index downward. Or said differently: all values, starting from the one that needs to be deleted, get the value of the next item in the array. After that, the size is decremented. The figure at the bottom should make things more clear.



making Lists with Local Arrays

By Bart Teunis

It looks like this in gml:

```
with (argument0)
{
    var i;
    for(i=argument1;i<size-1;i+=1) /*argument1:
position to delete*/
    {
        array[i] = array[i+1];
    }
    size -= 1;
}
```

Retrieving a value from a list will be necessary, too. This can be done with the `ds_list_find_value` function:

```
return argument0.array[argument1]; /*argument1:
position of the value*/
```

And that concludes the basic functionality for lists. For some other scripts to work with lists, you can have a look at the completed scripts package that can be found here for GM7: [lists.gmk](#) and here: [lists.gm6](#) for GM6. The maximum size of a list is 32000 items. A small disadvantage of GM6 is that when you use `ds_list_...`, it gives the error: "This functionality is only available in the registered version." That's why the scripts in the gm6 file all start with `ds_lst_`. In GM7, the scripts are executed instead of the built-in functions with the same name.

Completing the Package

Now the scripts do what they should do, but you also need to add the object each time. It would be nicer if everything could be added all at once. Time to complete the package!

Open a new GM file and add all the scripts to it. Also add an object called `obj_list`. Don't add anything else. Save the file as `lists.gmk`. Now you have yourself a nice "extension package" (not one like the ones GM uses, of course). If you want to add the `ds_list` functionality, simply merge the `lists.gmk` (or `lists.gm6`) file with your game and you're ready to use lists.

Conclusion

GM has functions to work with lists, but they are for registered users only, the scripts and implementation that we have done works for all users. This article has explained an implementation of lists by using local arrays. The method used here can also be used to add other types of data structures.

Become a writer at Markup Magazine

[CLICK HERE](#)

EXPANDING GM WITH .NET

By Sam Whited

If one were to browse a website such as the Game Maker Community (GMC), never before having used the .NET Framework, one might assume it was a “bad” practice¹ or would be immensely slow². However, this is not always the case. Granted, the .NET framework is slower than many other similar API’s, however, that speed decrease comes with many bonuses.

The .NET Framework is not lightning fast, but programming with it is. The .NET Framework is an immensely easy platform to build upon, and for larger projects it can save the developer a great deal of time. For smaller projects, native code may be the way to go, however, for large scale projects, the .NET can be a valuable tool. Unfortunately, Game Maker can only interface with specially constructed assemblies. Managed class libraries (like most of those in the .NET Framework) are often impossible to interface with from Game Maker. However, it is possible to work around Game Makers limited calling schemes and implement our own managed class libraries from within Game Maker with very little hassle. First I will outline some of the pros and cons of implementing the .NET Framework, and then I will discuss several ways to utilize .NET libraries from within Game Maker and explain the benefits and problems with each solution.

What you will need?

- The Microsoft .NET Framework 2.0 or higher
- A copy of Game Maker 7.0 Pro or higher
- Visual C++ and .NET knowledge
- Visual Studio 2005 Express or compatible software
- A copy of [The Cool Gamer's .NET Layer](#)

Architecture of .NET

To understand what the differences between native DLLs and .NET DLLs are, one must understand a bit about the architecture of the .NET Framework. .NET is a language-agnostic programming model, that is, it can be utilized from any language for which a compiler which supports it exists. To use .NET you must use a special compiler because most code written using the .NET Framework does not compile down to machine code; instead, it compiles down to a platform-independent language known as the Common Intermediate Language (CIL). The CIL is then compiled at runtime (known as JIT, or Just in Time, compilation) down to machine code by the platform-specific Common Language Runtime (CLR). This entire process is facilitated by yet another three letter acronym: the CLI or Common Language Infrastructure.

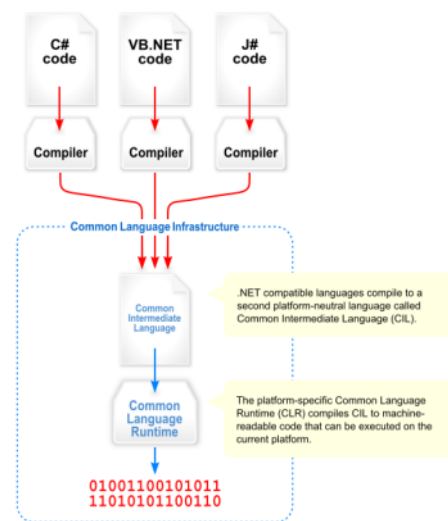


Figure 1

EXPANDING GM WITH .NET

By Sam Whited

Assemblies

After your code has been compiled down to CIL code, it must be stored somewhere that the CLR can find it to create machine code. That location is called an assembly and, by specification, that assembly must be a portable executable (PE) file (Namely a DLL or EXE). In this case, the assembly will be a DLL.

The Good

The benefits of using .NET class libraries instead of native DLL's are many. First of all, the development process is much easier as many common methods - hashing functions, file I/O, etc. - are already defined for you. The .NET API is extensive, and can be put to good use. Secondly, if you are using some version of Visual Studio, .NET DLL's are much easier to maintain and update than native DLL's. Also, code compiled to the CIL is platform-independent so long as a version of the CLR exists for the platform on which the developer wishes to deploy his or her assembly. And finally, CIL assemblies are verified for safety giving them better security than their native counterparts.

The Bad

Of course, every rose has its thorns. Because .NET DLL's are not reduced to machine code at compile time they often slower than native binaries. However, modern computers are so fast that this speed difference is negligible in most cases and if you are that worried about

speed you are better off not using Game Maker than you are not using the .NET Framework. Therefore this point is really a mute argument. It all comes down to what you want, speed for your users, or easy programming for yourself.

The Ugly

The .NET Framework is designed to run solely on Windows (Unless you use the shared source CLI, however, its licensing is very restrictive and does not allow for commercial projects). This means that the Mac version of Game Maker, or versions emulated on other systems, will most likely not be able to utilize your extensions created with .NET DLL's. If you are looking for cross platform interoperability, best stick with native code.

Using the GM .NET Layer

The methods of the .NET Framework are encapsulated in classes and namespaces which are inaccessible from within Game Maker. However, Game Maker can access managed global functions which in turn can access namespaces, create classes, etc. The Cool Gamer has provided an easy way to do this using his GM .NET Layer³. The GM .NET Layer resides between (you guessed it) Game Maker and the .NET Framework. To use the GM .NET Layer to create an instance of a managed object, only a few simple function calls are required. First of all, one must initialize the .NET Layer by calling `layer_init()`. This only has to be called once, before any other function. Next you can load an assembly by calling `layer_loaddll()`.

EXPANDING GM WITH .NET

By Sam Whited

This function will return a unique ID that can be used when referencing the assembly. Finally, one can create an instance of any given object using the function `layer_createinstance()`. Once an instance of an object has been created, we can begin to access its member functions.

Calling member functions of an instance is quite simple and only requires one function call to `layer_function()`. However, there are a few restraints on what sort of functions can be called. These restraints are not the fault of the .NET Layer; rather, they are built into Game Maker. All arguments and return types must be one of the following types: (System.) String, or (System.) Double. Also, if you want to call a function, it must be public and cannot be static. And finally, the function can have no more than 8 arguments.

Using .NET

While the GM .NET Layer is a great way to implement a few functions it is more of a workaround than anything else. Also, if you do not have access to the source of the assembly from which you are calling functions, the GM .NET Layer does not provide a way to suppress asserts and may throw long debug traces which you do not want your end users to have to deal with. In this regard, it may often be advisable to create your own wrapper layer for your individual project, or, if your project is going to be used solely from Game Maker, tailor it specifically for Game Maker so that a wrapper function is not even necessary. The first approach has the benefit of being able to access the .NET library without any structural modification to the library. This means that, since you will be accessing the library using C++, you can utilize functions which are encapsulated within classes and namespaces. It also

means that you can call functions with all manner of return and argument type just so long as Game Maker only gets a double or a string. The second option requires that you have access to the source code of the .NET DLL which you wish to utilize.

Game Maker can access global functions which return either doubles or null-terminating strings, therefore, you can simply create global functions which are exported from a DLL and can use the .NET all they want. If all of your functions are exported, you don't have to worry about namespaces or classes (which can still be used to store your data). Hereafter this article will require basic knowledge of C++ and .NET. It is also assumed that you know how to create a DLL for Game Maker, how to export a function, etc. For more information see my article in the [August 2007 issue of MarkUp](#) on creating DLL's for GM.

Loading an Assembly

If you choose the first option, you will most likely want to load an assembly for use by your function. There are several ways to do this: first and foremost is to simply select the assembly as a reference if you're source code is part of a Visual C++ project. To do this from Visual C++ Express click on the "Framework and References" section under "Common Properties" in your projects properties dialog. From this location you can add or remove references which will be utilized in your library. If you are not using Visual C++ (or if you would rather reference your libraries from code) you can also use the `#using` directive just as you would for any other DLL... So for instance, if I wanted to link with a DLL called "MyDll.dll" from my project I could either browse for it as described above, or I could place the compiler directive `#using "MyDll.dll"` in my code. The two methods are equivalent. There is, however,

EXPANDING GM WITH .NET

By Sam Whited

a third method which we may want to consider, and that is dynamic linking at runtime. The project properties and `#using` are compiler time directives. They are simply sets of instructions which tell the compiler how to behave. This means that they are only “executed” at compile time, after that they cannot be changed. However, for a DLL like the GM .NET Layer (which takes an arbitrary DLL and performs some action on it) we do not know the path to the library we wish to link to until runtime. For this we must use a useful little class buried deep in the .NET Framework’s API called “Assembly”.

The Assembly class does exactly what it sounds like it does; it holds information about an assembly. It can be located in the System.Reflection namespace which means that at compile time you will need to reference System.dll using one of the two methods described above. Once you have access to the System namespace, you can simply create a pointer to an instance of Assembly and set it equal to the result of one of the many static functions provided in the Assembly class which act as pseudo-constructors for Assembly (the default constructor simply allocates memory). These static member functions include the following:

- `Assembly.Load();`
- `Assembly.LoadFile();`
- `Assembly.LoadFrom();`

This means that to load an assembly at runtime all we have to do is call one of the many overloads of these functions; so for instance, listing 1 is valid code:

```
using namespace System::Reflection;  
Assembly^ a = gcnew Assembly::LoadFile("MyDLL.dll");
```

Listing 1

Calling Functions

Once you have your DLL referenced, you can use it however you would like. You can create and destroy instances of objects, access namespaces, call functions and member functions, etc. and, what’s more, you can define and export global functions so that Game Maker can do all this as well. Below is the source code for a simple DLL which uses the .NET Framework to display a message box.

Distributing Your Library

The final thing you have to remember before you allow people to download your extension is this: you are building your creation on a computer which has all of your projects dependencies already installed. However, your end user may not have any of the DLL’s your project relies on. This means that you also have to make sure that they get everything they need. Since you are using the .NET Framework it is quite obvious that your users will need to download it if they don’t already have it. To this end, Microsoft supplies a redistributable package which may be found on their website. This is a compact installer which you are free to include with your projects (look for a link in the resources section at the end of this article). There are several other dependencies which your project might require that aren’t quite so obvious, however. These include components such as the MFC and ATL libraries, and the VC Runtime Components. While these dependencies can be statically linked into your libraries, I would recommend packaging them with your library as DLL’s. To check what files are licensed for distribution, you

EXPANDING GM WITH .NET

By Sam Whited

can look at the file `redist.txt` located in your Visual Studio installation directory. To find out what files your assembly is dependent on, you can use the `dumpbin` utility installed with Visual Studio. To do this load the Visual Studio command line and call the following: `"DUMPBIN /DEPENDENTS [file]"` where `file` is the path to your assembly (for instance, `'MyDll.dll'`). Figure 1 is an example of `dumpbin` displaying the dependencies for a DLL called `GiiMote`. For simple DLL's the only dependency will be `mscorlib.dll` (which is a part of the .NET Framework and does not need to be redistributed individually), or `KERNEL32.DLL` (which is a part of Windows), and will not need to distribute other packages. However, if the need arises, make sure you know the license of each assembly you wish to redistribute.

Conclusion

Using the .NET Framework, the CLR, and managed C++ code or another CLI language in your applications may seem like a hassle, but in the end it is a truly rewarding

Listing 2

```
// If using Visual C++, simply reference these DLL's
// In your project's properties dialog.
// #using <System.dll>
// #using <System.Windows.Forms.dll>
using namespace System;
using namespace System::Windows::Forms;

#define exp extern "C" __declspec( dllexport )

exp double gmnet_show_message(char* message)
{
    System::String^ m = gcnew
    System::String(message);
    MessageBox::Show(m);
    delete m;
    return ( 1.0 );
}
```

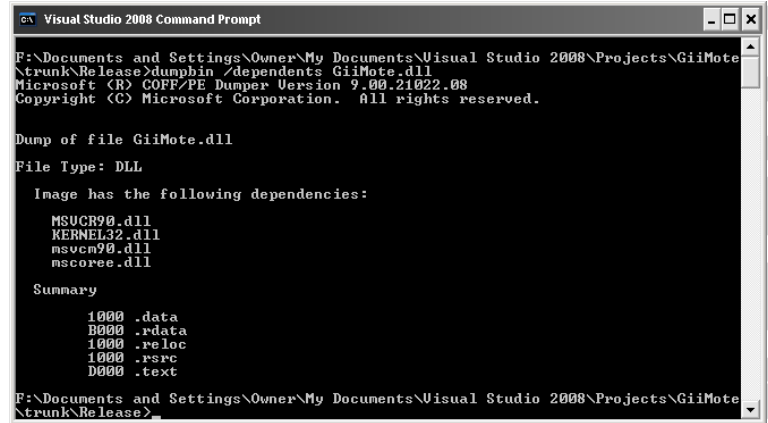


Figure 2

experience. The .NET can expand Game Maker further than many languages by easily implementing features such as reflective programming, memory streams, high-level operating system access and low-level hardware access and in many other ways. It is a general interface which encapsulates methods for nearly every situation. It generalizes programming languages, bringing them closer together and bridging the gaps between them. It is a brilliantly engineered collection of software which, when used properly, can be more efficient and user friendly than code written without it. The .NET API's make development a breeze and are sure to save you a great deal of time and effort, so fire up those compilers, add the `/clr` switch to your command line, and explore the possibilities of Game Maker and .NET.

Endnotes

- ¹ [Uuf6429](#)
- ² [Uuf6429](#)
- ³ [Link](#)

2D ARRAY STRING EXPLOSION

By Eyas Sharaiha

Back in Issue 4 of MarkUp Magazine, I explained how we could use GMLScripts.com's 'explode_string' function in order to parse CSV files. Recently, I received a request asking me to write an article about exploding strings into 2-dimensional arrays, and that is why today, I present to you explode_string_2d().

Background

For those who do not know what explode_string is, it is a function (popularly found in PHP) that separates a string into different substrings located in an array. They are separated according to a "separator character", which – when present – causes the function to split the string into one further 'splice'.

For instance, the string:

```
"1997,1998,1999,2000,2001"
```

If such string was split so that "," would be the separator character, the result would be an array with five values at [0...4] as shown below:

Index	Value
0	1997
1	1998
2	1999
3	2000
4	2001

2-Dimensional Arrays

Now, what if we needed to parse a string into a 2-dimensional array according to two separators? For instance, if we had the string:

```
"year,car,cost;1999,Ferrari,200000;1999,Bentley,250000;2000,Ford,35000;2001,Hyundai,15000;2001,Kia,17000"
```

If "," was the x-separator string and ";" was the y-separator

string, then we should get the following result:

Index	0	1	2
0	year	car	cost
1	1999	Ferrari	200000
2	1999	Bentley	250000
3	2000	Ford	35000
4	2001	Hyundai	15000
5	2001	Kia	17000

Parsing such string into a two-dimensional array could surely be done in a multitude of ways. One way that might occur when you first think about the issue is the construction of 'two loops', one that parses the different "rows" of the array first, and then another loop embedded in that loop that parses each 'row' individually. However, such method is rather time and resource consuming; for every row that exists in the newly-created array, an entire loop needs to be performed individually again – this time on the row in seclusion – to actually parse it.

When you think about it, a lot of 'duplication of tasks' would be going on; the entire string was already read to parse it into rows, why do we need to read the separate rows all over again to parse them into fields?

My Method

The method I prefer to use requires the string to be read only *once*, **character-by-character**, and placing the read field into the respective row or column according to number of 'spotted' x- and y-separator characters before that.

2D ARRAY STRING EXPLOSION

By Eyas Sharaiha

The Script

```
/* explode_string_2d(Data string, X-separator string, Y-
separator string, Array_name string);

Eyas Sharaiha, MarkUp Magazine*/
var data,sepx,sepy,array,_X,_Y,_POS,_LENGTH;
sepx=argument1;
sepy=argument2;
data=argument0;
array=argument3;
_X=0;
_Y=0;
_POS=1;
_LENGTH=string_length(data);
if
(string(string_char_at(data,_LENGTH))!=string(sepy))
{
    data+=string(sepy);
    _LENGTH+=1;
}
current_field="";
repeat(_LENGTH)
{
    CHAR=string_char_at(data,_POS);
    if(string(CHAR)==string(sepx)) then
    {
        //save previous string
        variable_local_array2_set(string(array), _X, _Y,
string(current_field));
        current_field=""; //reset current field data
        _X+=1; //move to next field
    }
    else if(string(CHAR)==string(sepy)) then
    {
        //save previous string
        variable_local_array2_set(string(array), _X, _Y,
string(current_field));
        current_field=""; //reset current field data
        _X=0;
        _Y+=1; //move to next row; reset column count
    }
    else
    {
        current_field+=string(CHAR);
    }
    _POS+=1;
}
```

Explaining the Script

First, you can see we have four arguments:

Argument	Type	Description
0	String	The data string in which the set of character-separated data is stored.
1	String	The x-separator character. Must be a single character otherwise the script wouldn't work. This separates different fields within a row.
2	String	The y-separator character. Must be a single character otherwise the script wouldn't work. This separates different rows within the 'data string'.
3	String	The name of the array as a string. It must be a local array.

One convention needs to be noted, and that is a 2D array has two indices and is written in the following way:

```
Array[ind1,ind2]
```

Both **ind1** and **ind2** have real values. While physically, each index doesn't correspond to an actual dimension, it is often assumed that the first index is the x-value and the second index is the y-value of the array; therefore, **ind1** corresponds to the row while **ind2** corresponds to the column.

The required variables are initialized in the script at first.

Then, the following lines of code are written:

```
if (string(string_char_at(data,_LENGTH))!=string(sepy))
{
    data+=string(sepy);
    _LENGTH+=1;
}
```

These are really just a couple of lines that make the script more flexible in reading the string. For instance, if we assume that a comma "," separates the x values and a semicolon ";" separates the y values, then a semicolon is needed at the end of the script. In order to make such placement of a semicolon (which will often be forgotten) *optional*, these lines check whether or not the final

2D ARRAY STRING EXPLOSION

By Eyas Sharaiha

character is the y-separator, and if it isn't, the y-separator is added (and this is reflected in the length of the string, since a character is added).

The variable `current_field` will be the variable that temporarily stores the field that is being read (yet not completely read). **After** the script finds a separator (either an x- or a y-separator) the data in `current_field` will be copied to a certain location in the given array, and the variable `current_field` will be emptied.

A (single) loop is then performed that repeats itself on each character in the string. The character is then read using the function:

```
CHAR=string_char_at(data,_POS);
```

Now, to check the type of character present (a normal text character or a separator character), we get a series of if statements (**Note:** the if-else statements could be substituted by a switch-case statement, alternatively).

```
if(string(CHAR)==string(sepx)) then
{
    .....
}
else if(string(CHAR)==string(sepy)) then
{
    .....
}
else
{
    .....
}
```

The first statement assumes there is an x-separator present, while the second assumes a y-separator is present. If both checks fail, the third block of code is executed.

If an x-separator is present, the previous string that was temporarily saved at `current_field` will be saved (as the x-separator marks the end of a field in the middle of a row) and then emptied. Afterwards, the current X index value will be incremented by 1, since the next field will start

being read next.

If a y-separator is present, the previous string that was temporarily saved at `current_field` will also be saved, since the y-separator marks the end of a field at the end of a row, *as well as* the end of the actual row. Afterwards, the Y index will be incremented by 1 since the next row will start being read. The X index value will be set to zero since the next row must be read from the beginning and therefore the first field in that row must reflect that.

If no separator character is present, the character `CHAR` will be added, as is, to the temporarily `current_field` variable so that it would be stored later on in a field.

If you want to use a switch-case statement instead of a series of if-else statements, then it should look like this:

```
switch(string(CHAR))
{
    case string(sepx): ...;break;
    case string(sepy): ...;break;
    default: ...;break;
}
```

After each loop ends, the `_POS` variable is incremented by 1, allowing the script to read the next character afterwards.

Conclusion

Such script needs to be run only once and will completely parse the string into a 2-dminensional array of your choice. The script parses the string into a local array, but changing the "`_local`" into "`_global`" in the script will change the array from local to global. Hopefully in next issue, I'll emphasize on how such way of exploding strings into 2d arrays can create a CSV reader that is **up to the CSV specifications** completely.

The making of FALLING TROY

By Alaric Holberton (Rikrok)

Getting started with design choices

When I saw the Ancient Civilizations competition up on Yoyo I decided straight away I wanted to enter. I'm a real perfectionist and as a result I can take a long time to create something to my satisfaction, having a deadline would be good for me as I'd be forced to not be over ambitious, I knew I'd have to have a finished product in three months. For the past year or so I had off and on been working on another game, Zero Point, which ate so much of my time I considered scrapping it on a few occasions. That game taught me how to use Game Maker, and as importantly- don't try and make a game that would require a studio's worth of people by yourself. (I may finish Zero Point one day but for the time being people will have to make do with one level).



So bearing in mind I had a time limit I had to decide what kind of game to make. What I needed was to not have to make tons and tons of graphics. I like my games to look good but it takes a long time. I thought the solution for

having a high standard of visuals but not run out of time making them was to set the game's location to one place. The genre of game that would most suit just having one location would be a puzzle game. With a platformer or shooter the game mechanics are fairly standard, obviously it differs from game to game but the main variation comes from the worlds you explore and the enemies you encounter. Two adventure games may have identical systems but if the stories and environments are totally different, they are different games. With puzzle games however no one is impressed by a Tetris clone. Once someone has made Puzzle Bobble no one would be interested in someone else taking the idea and putting different graphics on it. I had to therefore think of an original concept.

It occurred to me in the majority of fast paced puzzle games you directly control the objects that need manipulating, where as in the majority of other games you control a character. I decided I wanted to combine the two. Something I always enjoyed in platform games was when the platforms you were jumping about on were moving relative to one another, it made the gameplay feel more skillful. These were the factors that made me settle on my game concept which was: different types of blocks fall from the top of the screen, you control a character that has to jump from block to block, the way you jump off them determines where they go, stack the same type in rows or columns, the more you get together the more points you get when they are cancelled by a different type of block landing on them. The game progresses by it getting faster the better you do, the ultimate goal: get a higher score than anyone else (using online high scores). The main challenge from the game comes from having the skill to move your character very quickly and from thinking quickly, not hesitating when things get hectic.

The last thing I decided before starting to make the game

The making of falling troy

By Alaric Holberton (Rikrok)

was its setting. Out of the ancient civilizations I knew most about the Egyptian, Roman and Greek so it made sense to go for one of them. I settled on Greek as I imagined the competition would be full of the other two. I didn't choose a title for quite a few weeks. At one point I was going to call it "Jumping for Troy", I cringed when I said it to people though, so I'm glad I scrapped that.

Programming

Character control

My order of making a game tends to be code, graphics and then sound, though I do mix that up a bit so I don't get bored of one thing, and obviously there's overlap between the three. I started with character control, the character being at that point a green circle. I think the ultimate 2D platformer gameplay is Mario; not sluggish nor inaccurately fast, subtle acceleration and deceleration, and control over how high you jump by how long you hold the jump button for. It was this last point that required some slightly unnatural code, simply because physically it's completely impossible- your downwards acceleration (upwards deceleration) should be constant, otherwise effectively you're changing gravity depending on how long you hold jump in the air. Still, it makes for a more fun game, and I've never quite got why people would choose realism over fun in a game (unless you're talking Gran Turismo, in which point making it arcadey would destroy the point of the game, but then, I'd choose Burnout over GT). Anyway, the way I dealt with this was:

Provided there's nothing underneath our man, increase his `vspeed` up until a certain maximum falling speed. When you press jump (and you're standing on something) don't simply momentarily set `vspeed` to a negative value but instead as long as that jump button is pressed have a timer increase, and set your `vspeed` accordingly. It's very simple code but many GM games miss it. Different values for

different times means you can still incorporate a decrease in upwards speed but not have it nearly as much as your usual gravity.

```
// Jumping
if (jumping == 1) {
    jumptime += 1;
    if (jumptime <= 5) {vspeed = -17}
    if (jumptime > 5) {
        if (jumptime <= 10) {vspeed = -12}}
    if (jumptime > 10) {
        if (jumptime <= 15) {vspeed = -7}}
    }
```

A little note about using "if..." within another "if...", unless you have to use an "else" statement afterwards it's much better than using "&&" because if the first statement is not true the computer doesn't have to read anything else. It may seem small but over the whole code of your game it does make a sizable efficiency gain. My current pc is a nice 3GHz dual core, but I've kept my 2.4GHz Pentium 4 as a testing machine. I aim to keep everything running smoothly on that, but it's also very useful as I can see efficiency gains more clearly on it. Using Task Manager the CPU usage of the game might vary 20% on the old pc and only a couple of percent on the new one.



The majority of the character's code is fairly straight

The making of falling troy

By Alaric Holberton (Rikrok)

forward, though it may be useful to show how I got him descending smoothly on the platforms of various speeds.

```
// Landing
if (vspeed > 0) {
    if (!place_free(x,y + vspeed)) {
        move_contact(270);
        if (distance_to_object(obj_wood) <
            distance_to_object(obj_metal)) {
            if (distance_to_object(obj_wood) <
                distance_to_object(obj_stone)) {
                vspeed = (instance_nearest(x,y,obj_wood)).vspeed}}
            if (distance_to_object(obj_metal) <
                distance_to_object(obj_wood)) {
                if (distance_to_object(obj_metal) <
                    distance_to_object(obj_stone)) {
                    vspeed = (instance_nearest(x,y,obj_metal)).vspeed}}
            if (distance_to_object(obj_stone) <
                distance_to_object(obj_metal)) {
                if (distance_to_object(obj_stone) <
                    distance_to_object(obj_wood)) {
                    vspeed = (instance_nearest(x,y,obj_stone)).vspeed}}
            }}
    }
```

So this says if we're going downwards and the point where we'd be in the next frame is not free, move downwards (direction 270) to where we hit that object, and then take that object's speed. We're only interested in three objects, and to make sure we take the right one we find which is closest.

One other thing I'd say is that if you're making a game that isn't mouse based add gamepad support. It's simple to do and can make such a difference, and lots of people have an Xbox 360 controller or something similar connected to their pc. Also for 2d games the dpad tends to be better suited than the analogue stick, it's faster to switch directions as your thumb has to move a lot less.

The Blocks

While obviously there are several objects in the game, other than the man the main ones are the blocks of metal, wood and stone, which are all essentially the same, and an object that deals with two arrays that tell everything what

is where. Each array is 3 by 29, 3 for the 3 columns, and 29 for the number of blocks that can stack into them. For every frame it is checked whether a block is present in one of those slots or not, and then the corresponding array value is set to metal, wood, stone or nothing. The first array only recognizes a slot being filled if the object is stationary, i.e. the block has come to rest and it can then be determined whether we've just added to our wood stack or put something else down and need to cancel the whole lot underneath. I'm only showing less than 1/9th of what this "for" statement does as it gives the idea.

```
// Stack
for (z=2; z<=30; z+=1) {

    // Stack accurate
    if (instance_position(265,608-(z*20),obj_wood)>0) {
        if ((instance_position(265,608-(z*20),obj_wood)).vspeed == 0) {
            if ((instance_position(265,608-(z*20),obj_wood)).hspeed == 0) {
                global.ar_stack[1,z] = 'wood'}}}
    }
```

This shows in column 1 (x position 265) checking from 2 up to 30 (1 is the floor) and y position corresponding to the slot (each slot is 20 pixels high) firstly whether there is a wood block, then whether that wood block is not moving either vertically or horizontally. If all this is true then we can set that array value to be wood.

The second array is not so fussy about whether the block is moving or not. This is used for sending blocks into columns at the right place. Say the stack of blocks on your right is 20 high and you're standing on a block lower than that and you want to send it up to the stack on the right. The block needs to know how high it should travel to be at the top of that stack. If we just determined the height of the stack using our first array there could be a block a couple of pixels away from landing on that stack and we'd send our block straight into it. So we have a second array that will include blocks around the top of a stack, but that are not necessarily motionless. The code for this is more or less the same, except obviously there are no speed == 0 conditions and it checks at two different y positions each

The making of falling troy

By Alaric Holberton (Rikrok)

time that are 15 pixels apart from one another to cover a larger vertical distance. Finally to deal with the unlikely event that two blocks do settle overlapping each other; every stationary block checks that there isn't a block sitting in its position with a different id. If this happens then the block simply jumps up by 20 pixels.

There is a lot of code for each block to determine whether its status is normal, flashing as part of a stack in a row and/or column or needs to be deleted, so I won't copy it all in but instead explain the basics. Using the arrays it's fairly easy to tell what's around, what ended up being more complicated was making sure the blocks belonged to the right group. It was not enough to set a variable for a metal block to be flashing, and then when the block gets cancelled out count how many were metal and flashing, add to the score and delete them, as there may be two or more separate stacks of metal. What I did was when a new group was formed (three of the same types of block were together) those blocks take a variable (in fact I used another array outside the blocks to store these) that is equal to the id of the third block. Any additional blocks joining see there is a value present and take it too. This way whenever a block in a group is touched by something else and needs to be deleted it sets a global variable (checkdelete) to 1 along with the group id. If any block has that value it knows to destroy itself. The only other thing to bear in mind was if there were two stacks connected by having three in a row across and that row got broken new values had to be assigned so they'd be seen as two separate groups again.

The Tutorial

Many games will want to have a tutorial showing how to play, and just having text is quite a boring way of doing it. What I wanted was to also have the character perform whatever the text was saying, it's more interesting and demonstrates things more clearly. It would have taken me forever to type in where to move when, so instead I coded

things so I could record myself playing and then play that back. In the step event of the record object I used the following code:

```
ds_grid_add(grd_record,0,global.tutorialtime,global.left);
ds_grid_add(grd_record,1,global.tutorialtime,global.right);
ds_grid_add(grd_record,2,global.tutorialtime,global.down);
ds_grid_add(grd_record,3,global.tutorialtime,global.jump);

if (global.tutorialtime = 2300) {
    file_tutorial = file_text_open_write('data\tutorial');
    for (i=0; i<=2300; i+=1) {
        file_text_write_string(
file_tutorial,string(ds_grid_get(grd_record,0,i)) +
string(ds_grid_get(grd_record,1,i)) +
string(ds_grid_get(grd_record,2,i)) +
string(ds_grid_get(grd_record,3,i)));
        file_text_writeln(file_tutorial);
    }
    file_text_close(file_tutorial);
}
```

So every frame a grid records a 1 or 0 for the four buttons I could be pressing. At the end it is all written to a text file. To play it back in the create event I have:

```
grd_play = ds_grid_create(4,2300);
file_tutorial = file_text_open_read('data\tutorial');
for (i=0; i<=2300; i+=1) {
    var_line = file_text_read_string(file_tutorial);

ds_grid_set(grd_play,0,i,real(string_char_at(var_line,1)));
ds_grid_set(grd_play,1,i,real(string_char_at(var_line,2)));
ds_grid_set(grd_play,2,i,real(string_char_at(var_line,3)));
ds_grid_set(grd_play,3,i,real(string_char_at(var_line,4)));
    file_text_readln(file_tutorial);
}
file_text_close(file_tutorial);
```

The code above creates a grid and puts all the values from the text file into it. Then in the step event simply:

```
global.left = ds_grid_get(grd_play,0,global.tutorialtime);
global.right = ds_grid_get(grd_play,1,global.tutorialtime);
global.down = ds_grid_get(grd_play,2,global.tutorialtime);
global.jump = ds_grid_get(grd_play,3,global.tutorialtime);
```

Usually `global.tutorialtime` increases by one every

The making of falling troy

By Alaric Holberton (Rikrok)

frame, though it will pause and wait for the player to press a button when prompted to. In the game the player's keyboard or gamepad's input determines `global.left` etc. but clearly this needs to be disabled during the tutorial.

Online Scoreboards

Having this was important to me as it adds the competitive element which makes people want to play more. I think it's worth mentioning because when I was first trawling through the forums for information on how to do it I found very little useful advice. The way I did it was as follows:

- Get a web host that supports MySQL and php.
- Set up a MySQL database using your host's control panel. They should have phpMyAdmin; use that to create your table with fields for name, score, whatever you want, maybe timestamp and IP address too.
- Make a php file for receiving the scores and another one for sending them. I thought about putting all the code in but then this article would start to get very long. You can learn it all very easily at w3schools.com/PHP/php_mysql_intro.asp
- Upload these files to your server. Check it's all working fine by pointing your web browser to the first php file, in phpMyAdmin put some stuff in your table and it should come up when visit the php file. You can test your second one is uploading ok by going to `www.yourwebsite.com/game2.php?name="Jim"+score="200"`.
- In Game Maker use 39dll (I originally used GMSQL, but this is less secure as instead of using the php files you connect directly sending the password, it's also a lot slower). Follow the tutorial that shows you how to connect to a web page. Instead

of Google access your php file. In order to get your bunch of names and scores into a useful format in Game Maker (rather than a long string) do something along the lines of:

```
setformat(sockId, 2);
while(1) {
    size = receivemessage(sockId, 6000);

    if(size > 0) {
        readsep(" ")
        for (j=0; j<=49; j+=1) {
            ds_grid_set(global.grd_scores,0,j,readsep(" "));
            ds_grid_set(global.grd_scores,1,j,readsep(" "));
            ds_grid_set(global.grd_scores,2,j,readsep(" "));
        }
    }
    else break;
}
```

To add some security I also wrote my own encryption for the scores, though of course as it is written with Game Maker if someone really wants to hack then they could.

A final note on the scoreboard; I wanted to add the player's country as well as name. I think it makes it more interesting than just a list of names, to see where in the world people are playing and players may enjoy getting their country ranked high. It took quite a while collecting all the flags and standardizing them but at least I can reuse the graphics and code in other games. A script simply runs and draws the appropriate flag for the ISO (International Organization for Standardization) two letter country code it's sent. I have 221 countries out of 246 which isn't bad, though I was worried about causing offence for those missed, however they tend to be very small countries and it's unlikely someone from there will play. Generally I stuck to the ISO list, though I included England, Scotland, Wales and Northern Ireland for those who didn't want to just put Britain. It also led to some interesting reading about disputed and unrecognized states.

The making of falling troy

By Alaric Holberton (Rikrok)

Graphics

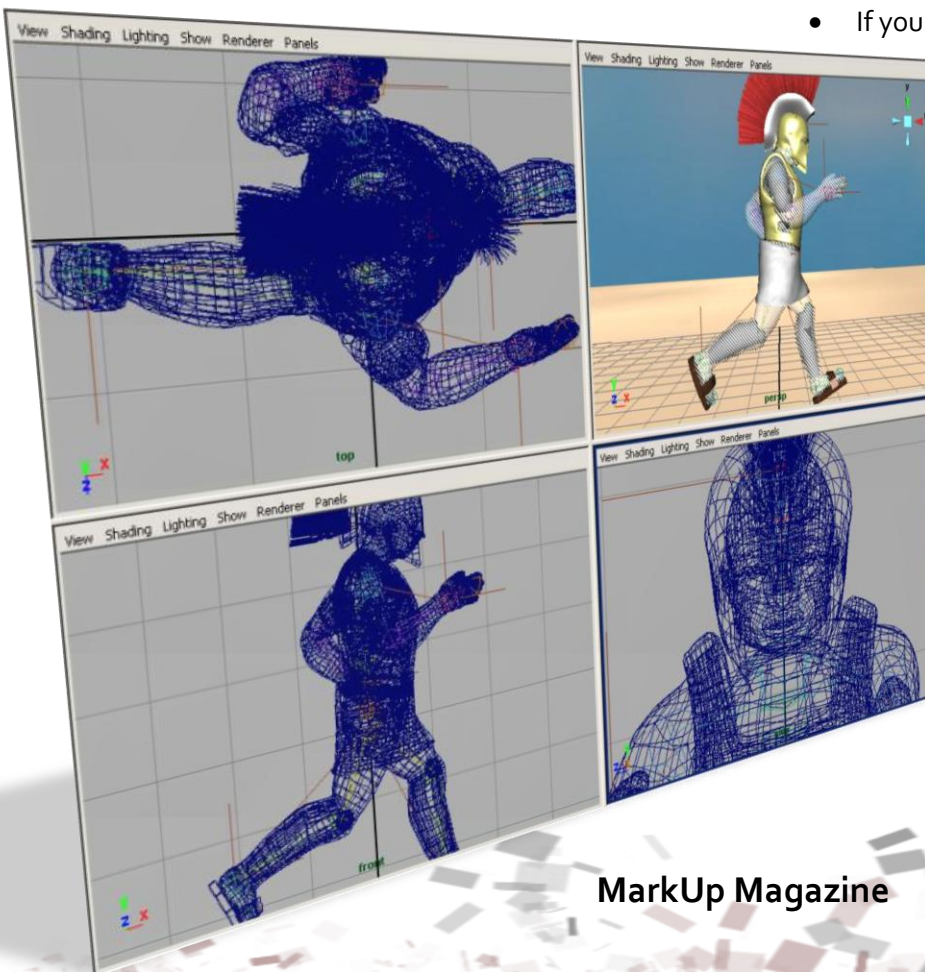
I imagine most people will find what I say here of more interest, I know I started to get bored coding all my array checking and correct grouping. I used Autodesk's Maya for the majority of the graphics. It's a massively powerful 3D modeling and animation program that's used in films such as Lord of the Rings and The Matrix, even Terminator 2 all those years ago, as well as of course many video games. Amazingly Autodesk offer a free Personal Learning edition that anyone can download which has most of the functions of the full version, the main limitation being that you can't use it for commercial projects. It's a big program to learn but it's also very satisfying. The other main 'competitor' (though Autodesk owns them both now) is 3D Studio Max, it just happened that I got started with Maya and it's what I know and like. I got into it as I produce music, and sending CDs off to

record labels I wanted to design nice sleeves and not be limited by Photoshop. If you're interested in learning Maya there are many tutorials available on the web and it's way too big a topic to get into, but I will give some advice to help get started and almost instantly get some impressive results.

- Use Mental Ray when rendering.
- The default shadows from a light are depth map, switch it to ray tracing.
- In Mental Ray's rendering options enable final gathering, it simulates light bouncing off surfaces without massive cost, your images instantly look much more real.
- You may want to set your camera's focal length quite high for making graphics for 2d games, this way you get rid of unwanted perspective.
- If you attempt to animate a humanoid the best

advice I can give is to stand up and do the motions yourself, usually in slow motion. You feel ridiculous but it's a great way to analyze movements so you can recreate them realistically.

Once I had rendered an image (use TIFF as it includes the alpha layer) I would open it in Photoshop (if you don't have it GIMP is a popular free alternative: www.gimp.org), makes any changes I wanted, save as a PNG and import in GM. Even though GM saves images regardless of what format you import them in using PNG makes the partially transparent pixels maintain their correct color, whereas with a jpg they would get them merged with white/another color. Unless you're going for an eight bit look I would always use alpha masks for smooth



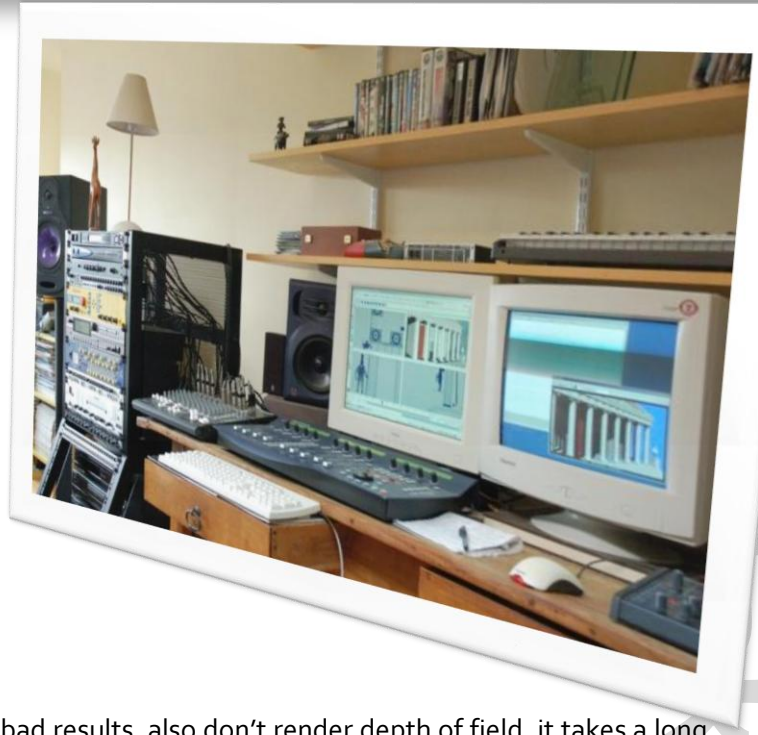
The making of Falling Troy

By Alaric Holberton (Rikrok)

edges. In Photoshop just do a color overlay of white on the layer's blending options and add a black background, then use `sprite_set_alpha_from_sprite` in GM. This gets infinitely better results than setting a background color then using GM's transparency option. If your game features a lot of graphics it may be wise to keep them as external jpgs to reduce your file size and then use `sprite_add` to load them in game. If you do this and are using Maya make sure you uncheck 'Premultiply' under Frame Buffer in Mental Ray's options, otherwise your partially transparent pixels will be blackened. You may want to use an encryption dll for the external files so people can't easily steal your graphics.

The other piece of software I used for the mountains in the background is Terragen 2 Technology Preview (www.planetside.co.uk/terrigen/tg2/index.shtml). Again it's free for non-commercial use and you can create stunning outdoor visuals within a few minutes. It's a very impressive piece of software and very easy to use.

I don't have a problem with highly saturated colors depending on the style of the game, but don't use too many different colors otherwise things look a mess and incoherent. Use parallax scrolling *everywhere*. If something is at a different depth on the screen it should move at a different speed, the further away the slower. Finally you may or may not want to use depth of field effects; blurring things in the distance or that are closer than where the action is. I used it a lot in Falling Troy as it tends to be visually impressive and stops you being distracted by backgrounds that aren't part of what you interact with. It did pain me though to lose all the detail on the Terragen mountains. Simply apply Gaussian blur in Photoshop, varying the amount depending on the strength of effect you want and how far away something is. If you rendered a background with visual depth to it and need varying blur, such as the temple in Falling Troy, you'll need to create a z-depth image (rendering z-depth information into a file format that supports it tends to get



bad results, also don't render depth of field, it takes a long time and is horribly grainy, post processing looks much nicer). Assign a surface shader of white to everything in your scene then add dark fog. You get a grayscale image where what's lighter is closer and what's darker is further away. Be careful with anti aliasing because it effectively creates false depth information, instead render at double resolution and down scale after your DoF processing. You then need a good depth of field plug-in for Photoshop. Surprisingly most of them are pretty bad; the only one I really recommend is Frischluft Lenscare.

Sound

I was in a good position for making the music and sound effects for Falling Troy as I've been producing music for quite a few years now and have a small studio's worth of gear. Things have changed a lot however; where once you had to buy expensive outboard equipment for everything now most can be done with software.

I started by searching for modern recordings of ancient

The making of Palling Troy

By Alaric Holberton (Rikrok)



Greek music to get an idea of it. Unfortunately I found very little, and what I did hear was pretty uninspiring. I decided to have some classical sounding strings and Greek guitar, but basically not try and keep things authentic, instead go for a chilled out Balearic House sound for a sunny Mediterranean feel. It was quite different making the music for the game as it didn't need to have the structure of a normal track, instead it would loop over the course of about ten minutes, which meant I could have lots of different stages to it, using more melodies than you could usually get away with. The music is more like a ten minute mini mix. I was tempted to make it longer but I had to keep file size in mind.

In GM I didn't use any of the inbuilt sound functions, but instead used the Super Sound System dll by tsg1zzn (gmc.yoyogames.com/index.php?showtopic=120034). It's simple to use, doesn't have any licensing to worry about and supports the only file format that I'm interested in-ogg. Ogg gives significantly better sound quality than mp3 at the same bit rate, and again it doesn't have any

licensing issues that mp3 has. There are few things to bear in mind with SSS though:

- If you have a file over a certain length it gives up playing it when it reaches its limit. I had to cut my main music file into two.
- The volume and panning functions are not logarithmic, a volume of 5000 (maximum volume is 10000) should sound like half volume, but in fact it's barely audible.
- It doesn't support effects such as reverb, delay,

flanger etc. This is usually not a problem for me as I can have the effect in the sound file and it will be higher quality and not use processing power.

However where it was an issue was for the noise when you slide down the side of a pillar or stack of blocks. The sound file stops playing whenever you stop sliding, but with no reverb at the end it would sound very unnatural. What I did to get around this was have a separate sound file which was just a reverb tail to that sliding sound. Whenever the sliding sound is stopped the reverb sound is triggered. (My wall still has the scuff marks from dragging a shoe across it to make that noise).

I won't go into music production techniques as again it's too big a topic, but I will give pointers for recording voice.

- Put as much sound dampening around where you have your microphone as possible. You generally don't want the voice to sound like it's in a bedroom. Putting up duvets and pillows and

The making of Falling Troy

By Alaric Holberton (Rikrok)

whatever soft things you can find will absorb the reflected sound so you record a dry sound lacking in spatial information. Then you can apply the appropriate reverb on the computer for whether the voice is outside, in a room, cave, whatever.

Don't have your mouth too close to the microphone otherwise you introduce the proximity effect where the bass frequencies are amplified.

Before applying reverb:

- Gate: removes background noise during pauses.
- EQ: get a pleasing tone to the recording
- Compress: not in the usual computer sense; compress the dynamic range of the audio. Google vst compressors. This is important trick for giving audio a professional sound and can help the voice sound clearer.

Final Thoughts

Overall I was pleased with how Falling Troy turned out. I think I managed to do something original which is fun to play. If I were to add more things or do a sequel:

- Bonus and bogus items such as speed boost, bombs, blocks that can only be got rid of with canceling of stacks beyond a certain size etc.
- Graphically I'd like to have some people in the background supporting you; I always liked that touch in Street Fighter 2. Also a day and night cycle, with a man coming to light and snuff torches on the pillars and temple. Some birds would be nice too.
- Some kind of combination system where if you cancel two groups within a second, you get a multiplier. The greatest example of this and what

is in my opinion the best puzzle game of all time (and it's not even very well known) is Magical Drop 3 on the NeoGeo.

- The biggest addition would be a two player mode, both cooperative and versus. The 'versus mode' would involve the better a player doing the more blocks fall on the other player's screen, along those lines I'd also like to feature a versus computer mode. What really held me back from this weren't just time constraints but the speed limitations of Game Maker.

As much as I love Game Maker I think I'll only have one more game come out on it, after that I'll be focusing on C++. GM was never designed for overly complicated projects but the more you learn to do the more you want to do, until you're limited by the program and not the other way around. My final GM game should be something you wouldn't expect to see from Game Maker though; I'll keep progress on "Sky World" posted on www.rikrokware.com.

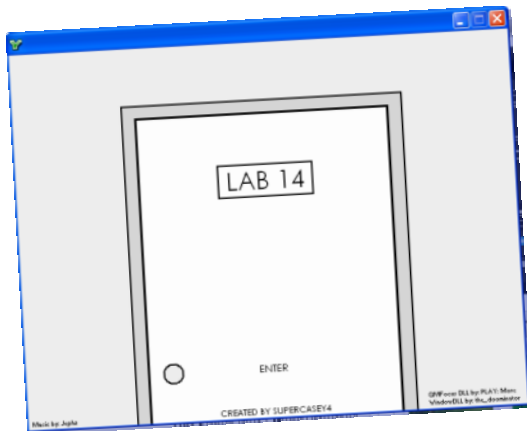


Lab 14

By erthgy

Review

Lab 14 is just one of those games that will confuse you, test your patience, diligence, skill at solving puzzles and finally provide a way to victory, in that order. It's kind of like Karoshi in the fact that it gives you a starting point... A small hint and a "you're off on your own pal" kind of challenge, then it's your expectancy to find that wonderful blinking yellow and red exit door(s, in some levels there is more than one exit).



Lab14's Nice-looking Title Screen

Probably the coolest part of Lab 14 is its requirement of dexterity, and if you can keep up with its fast paced puzzle solving, then you **are** the targeted audience for Lab 14. Sure, the game was amusing for the first few levels, but if you aren't skillful enough to keep up with the hard game play and no save button (for those of us that like to go back and play a level again, without playing through the whole game twice or more!) or for those of us that can't beat the game in **just** one sitting; then it is quite frustrating. Another thing that added onto this bad feeling of the game was only one background song throughout the whole game, and no sound effects! Heck, there's even no way to shut this song off (excluding the mute button on your PC, which isn't that hard to get to since this game is windowed)! So, it's not like this game isn't repetitive. The

game does deserve a bit of appraisal though, because the game play was the best game element (like in any high quality video game) in Lab 14 out of the other elements in the game; which means that SuperCasey4 did a good job in balancing the game.



This guy needs wings, a bungee cord, or just an old style "press and hold of the right arrow key"

Some improvements to this game would be to get a save system running, a level selector (like in Karoshi), differentiated background music (per level), sound effects, more game play elements, less hard puzzle solving, and Super Casey 4's got a good game! All in all, no one can deny that they liked Lab 14; it's just the fact that it's quite repetitive which will stiff arm non-puzzle gamers away.



See Next Page for Scores

Lab 14

By erthgy

The Verdict

Game play: Very repetitive, but not repetitive enough to be considered as disappointing.

Graphics: Highest quality, he could add some more graphical effects into the game to make it more appealing, however.

Sound: Background song gets very annoying towards the later levels of the game.

Controls: These controls aren't really that tough to get at all.

Replay Value: Quite bleak here, as you all ready have solved the puzzles, and with no save and load function, it gets very annoying.

Viscerality: Repetitive, but immersive in the beginning levels.

Overall: So Super Casey 4 has created a good game, but, with a few core improvements he will make it a **lot** better.

See Next Page for Interview

Quick Review

GNET

GNET is an excellent Game Maker Extension that uses the "GNET" DLL. The extension allows you to take advantage of the .NET Framework in Game Maker!

The extension supports a wide range of functions that are available in the .NET Framework; these include Math, a better file input/output mechanism, console windows, event logs, process and memory functions, dialog boxes, and more!

Of course, in order to be able to use the DLL, the .NET Framework must be installed on your computer. Note that the GNET DLL uses the .NET Framework 2.0 rather than the latest version, 3.5. You will therefore miss on some of the newer features of the .NET Framework, such as Windows Presentation Foundation, etc.

Get it now: gmc.yoyogames.com/?showtopic=286021

>>> Scores

Ratings

Gameplay

Graphics

Sound

Controls

Replay Value

Viscerality



Overall Score



Get it now!

yoyogames.com/games/show/30381

Grading Compared to Professional Quality. This means the game was good enough to be compared with commercial games in the market.

Interview with SUPERCASEY4

By erthy

Interview

What inspired you to create Lab14?

I got a lot of inspiration from Psychosomnium by cactus, and maybe a little inspiration from Seven Minutes by Virtanen. I had originally intended for the game to have more of a story and be more abstract, but as I made it, I realized that all of that was just a distraction and the game was more fun if all you had to do was get through the level. I got most of the ideas for the levels from just using the computer, and some of the others from looking through the manual for functions that could be interesting.

Interesting, so, what was the hardest part in making Lab 14?

Coming up with the puzzles, there really isn't any difficult coding in the game, and it only took me about a week to code it, but it took me much longer to come up with interesting puzzles and clues.

Cool, so, would you say that Lab 14 relies heavily on good room design?

I wouldn't say it relies on room design as much as making you think outside the box. A lot of the levels could just be an empty room with the clue in it, but in a lot of levels I tried to trick the player by putting something in the level that seems like the obvious way to do something, but it really has nothing to do with beating the level.

Yeah, I did notice that as I went through it, I don't mean to point this out to those that didn't play the game, but once I actually found "Press T", I was on the floor laughing from my lack of common sense (as in, not able to solve it

so simply), lol. Anyhow, let's get back to talking about Lab 14. What was your favorite part in making Lab 14?

I guess my favorite part would have to be finishing it. Usually I like coding something that I have never done before, but in this game there really wasn't anything like that. I mean, I've made plenty of platform engines before, and the graphics were pretty simple, so finishing the game and seeing people play it was my favorite part of making it. There's nothing more frustrating than making a game and having no one play it, but there is also nothing better than making a game and having people play it and enjoy it.

Will we ever see a sequel to Lab 14 like Lab 28 or something?

Well, I have an idea for a prequel which shows how you ended up in Lab 14, but I don't really have many ideas for the puzzles, so it might be a while before I make it. It probably won't be level based like Lab 14, but I don't really want to give too much of it away.

Last question, what did you learn from the experience, and what suggestions do you have for others?

I guess I learned that people are a lot better at solving puzzles than I thought (I thought people would have more trouble figuring out some of the puzzles, but it seems like it's fairly easy for some people) so you can expect the sequel/prequel to be more difficult. And for suggestions for others, just keep making good games and eventually one will catch on, don't get discouraged. And also, program your games at 60 frames per second; it makes them look much smoother and more professional.

GAME REVIEW

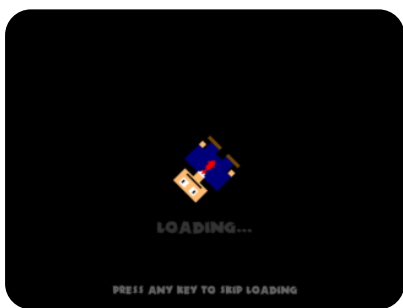
Karoshi 2.0

By erthgy

Review

I think that maybe the majority of people out there are highly unwilling to play puzzle games. When they first tried it, it was too hard, so they gave up. Sometimes it is the creators fault, **but not in this game**. This game gives you the perfect challenge in a simple and flat-out way that is very hard to explain, but I will do my best.

This game's menu truly reflects upon the game play and friendliness of the game. It features a level editor, built in extra levels, a "7 minutes" mode, a watch-the-blocks-stack type mode, and a way to share levels you created with friends who also have played the game. The idea of the game Karoshi is to kill yourself or go suicide, (whatever you want to call it, either way your objective stays the same). In some levels, players will find themselves plat forming their way onto spikes, and splashing their character onto the wall in a bloody mess. In other levels, players will find themselves being smashed from an object 10 times your size. Then finally my and most people's favorite, being killed from your own bullet's shot.



You can see by the look in his eyes that this is no normal platformer that copies Mario or some other original game idea.

This game just redefines puzzling as it is. Many people thought that some levels (such as level 4, 16, 11, 28 and the last one) to be too challenging. When I was going through this game, I understood what those players were saying; the game is quite repetitive at times, but it's fast paced advancing from level to level, and the feel of finally conquering that level, does make up for those deficiencies.

The only advice that I can give to those players that feel as if the game is too hard, is to go back to the level they were stuck on, and slowly think it through. Think of the simplest yet unobvious solution.



An easy to navigate menu gives for Replay Values

That also has to do with this next point I am about to make, the game goes very far against gaming laws, (and it does it absolutely correctly, too). Other puzzle games would never have that easy-to-solve, action based, rapid solving type feel that this game has. Another thing that other games wouldn't have is the sense of humor that goes along with accomplishing a level that you were stuck on. I laughed my head off once I found the solution to the riddle: "There is a key to this level". (I won't spoil this for you; it's a must-see-to-believe type of accomplishment).

Another cool thing I found in this game is the way that there are invisible walls on some levels, things disappear and re-appear once the player flicks a switch, pushes a block onto a button, or just plainly looks into a mirror, (this is on purpose though, so don't feel as if this game has coding problems.)

Some people believe that this game has no flaws, but now that I have explained to you some good points in the game, let us talk about the extremely few bad points. As I mentioned before, the game can get repetitive, (especially if you aren't smart and quick enough to solve some puzzles). The sound is a bit annoying to switch once you get into the quick thinking type puzzle solving I explained before, as the user has to do this manually from the game's menu. (However, I did NOT mention before that

Karoshi 2.0

By erthgy

this is quite simple and quickly done by just pressing the escape key and clicking on numbers 1-3 on the bottom right of the screen.) Lastly, the game does get a little glitchy for me and other people with slow computers, (I'm just saying this for 2Dcube's sake, but he should consider making a slower version, as many users of the GMC have quite slow computers).



As if this snapshot doesn't go against the gaming laws Miyamoto set for his cult-hit "Mario".

Ultimately this game gives everyone willing to think a fun and experiencing feel of accomplishment, which no gamer can deny. I just hope that other Indie game developers don't try to steal 2Dcube's awesome ideas, and turn them

into a badly put together game, but, that's for the future.

The Verdict

Game play: A little repetitive, but still awesome.

Graphics: Top-Notch, my favorite is the blood after your guy dies.

Sound: He needs to vary these up a bit, as these do get repetitive, but the sound is also cool.

Controls: At first, the controls were tough, but if you get used to them you'd understand why 2Dcube made them the way he did.

Replay Value: Replaying your all ready defeated levels in the 7 Minutes game play mode is extremely fun, and adds to the action feel of this game.

Viscerality: An Immersive experience, but it is repetitive.

Overall: Anyone should give this game a try, just be willing to try more than once. (The only reason it SEEMS I gave this game a low rating was because I thought deserved to be contrasted to professional games, which means I think that this game is awesome enough to go commercial.)

See Next Page for Interview

>>> Scores

Ratings

Gameplay



Graphics



Sound



Controls



Replay Value



Viscerality



Grading Compared to Professional Quality. This means the game was good enough to be compared with commercial games in the market.

Overall Score



Get it now!

yoyogames.com/games/show/32253

Interview with 20cube

By erthgy

Interview

What inspired you to create Karoshi?

I'm not completely sure. It may be that I saw the game "kill yourself in 5 minutes" which I believe is some sort of point and click game. I never played it, but saw a blog post of it. It may have been something else, but fact is the idea of a game where you'd kill yourself as a goal got in my head, and stayed there for some time until I decided to make a plat- former out of it.

Did you have any trouble on the game, or did it go quite smoothly?

I deliberately chose a simple type of game to make, since I dislike programming complex systems, physics, and so on. The thing I had the most trouble with was the electricity, and then I used it in only one level, lol.

How long hour-wise and day wise (as in, hours a day) did you spend on creating the game?

I'm not sure, since I started a few months ago. I had a

break where I didn't work on it for some time, and later I did half of the levels in the week before the release.

Will you ever add more plot to the game such as cut-scenes? Call me crazy, but I think it might work out quite well :).

It could work, but I have no plans for that.

What are some new ideas for your prequel/sequel coming out?

You can expect a lot more levels (in version 2.0). The puzzles this time are less about using logic to shove crates, and more about different solutions you'd normally not use in a game.

What was your favorite part of creating the game?

At some point I got in a "creative flow" of sorts, where I made about 10 levels in a row. That's when I made the K level and several others, and I kept working all night.

Write for Markup Magazine

[CLICK HERE](#)

Interview with 2DcUBE

By erthgy

What was your least-favorite (part of creating the game)?

There wasn't a lot of frustration in making the game, but I had a few minor problems with the electricity and the crates' physics.

Do you think that you will ever go commercial? Obviously you've already won the Yoyo-games first competition...

I study game design, so yeah, I intend to go commercial.

Do you have any suggestions for YoYo Games as they twist and update their beta?

Add those integrated high score lists please, and I think they should use more neutral colors for their site.

If you did go commercial, what console would

you make your game on? Or would you target PC gamers?

I don't really mind, but I prefer playing games on a couch and with a controller, so I guess consoles.

Who would be the Publisher of your game? Or would you just release it online?

I have no idea as far as publishers go. I like where it's heading with downloadable games and I think that might be the future.

Conclusion

2Dcube gave us a few good answers on game design; ultimately, you have to remember that game design is simple, creative, and most importantly fun!



>>>Future Dev

XML

The software world is moving towards XML in multiple ways. It is now becoming the primary way of data storage, as can be seen in settings files of recent games and even operating systems. XML is also popularly used in open document formats such as Open Document Foundation's ODF format and Microsoft's new OpenXML format as well as their new XPS (XML Paper Specification) format that competes with Adobe's PDF format. Not only that, but XML is also being used as an interface markup language as can be seen in Mozilla's XUL, in which applications or add-

ons like Chatzilla are almost completely made with XUL and run with the XUL Runner or a Mozilla Framework application. XML is becoming so popular due to its highly structured nature and extensibility. Currently, Game Maker only supports the inferior INI data storage mechanism natively, and XML is done through – often limited – external extensions, scripts, or DLLs. What Game Maker really needs is native support for XML that allows all GM developers to use this superior method of data storage for game settings definitely, but maybe even for things like level design, etc.

Eyas Sharaiha ■

GAME REVIEW A DRAGON'S TALE

By Shadow Master

Review

I always love reviewing a game which hasn't yet been released, and this one is no exception. Although this demo version is let down by some incomplete features and unpolished looks, it still makes for a very good game.

Around the Lizcren community, local fans have taken well to the latest instalment and are saying that the game is "Looking good." It has also been suggested that "the intro is way too long to not allow skipping; presentation was quite nice and the graphics were pretty good" and, personally, I must agree. It's no surprise that despite certain problems the game has gone down a treat.

As I have already mentioned, the demo is not exactly fault-free, the screen can be irritating, as it is small and quite cluttered, the opening sequence is very long, the game lacks a basic instructions/controls menu, I spotted some in-game glitches (including in the masking) and the names can be complicated for beginning players, which is not helped by seemingly unimportant details added to the game. But, keep in mind that this game is not yet released, and all of these problems are liable to change.

The features of the demo succeed in outweighing the faults, as the game has a truly original story, detailed sprites, interesting and well-developed characters, a large variety of different areas to explore, balanced and varying enemy difficulties and even the ability to switch between the characters in your party. These are just a few of the qualities that are to be enjoyed while playing the game.



In conclusion, I'm really looking forward to seeing this as a completed title, some slight tweaking, polishing and mending of faults would really finish this game off in an expert way.

>>> Scores

Ratings

Gameplay

Graphics

Sound

Design

Storyline



Overall Score



Get it now!

@64digits.com

Ancient Ants Adventure

By Shadow Master

Reviews

What do you get when you take a game with excellent graphics, fabulous gameplay, and a unique strategy approach, sprinkled with many other niches? I'll tell you. You get an utterly deserving competition winner.



Ancient Ants Adventure pits you in against legions of hostile creatures, such as flies, spiders and scorpions. It is up to you to command your ants with the ultimate goal of



Quick Review

Speech Dialog Extension

The Speech Dialog Extension by HaRRiKiRi forms an amazing addition to Game Maker. The extension allows you to load .dlg dialog files and use them for game interactions.

Such dialog files facilitate speech between characters in the game, where you can interact with another object in the game by choosing a statement to be said to the other character, to which the other character replies accordingly.

Different speech decisions trigger others to start and others to stop with the same character or even with others.

The extension allows for basic selection of speech items to be said as well as the dialog process to be displayed. While on the visual side the extension needs some improvement, it is definitely beneficial, especially for those of you who would like to make RPG or adventure games.

Get it now: gmc.yoyogames.com/?showtopic=288242

defending the colony.

The game begins with a brief opening message informing you of the current situation, before you lead a small troupe of warriors into the fray. Your team grows as you progress through the game and rescue other ants. You will then grow to uncover all sorts of new powers, to fit certain positions in your formation. But it will keep you hooked for a while before it is possible to say that you have beaten the game, with "Divine Tasks" on each level to provide bonus fun. Just when you think you're finishing the campaign is when the fun really starts!

It's no surprise that the game has gone down a treat in the community, being praised as an "awesome game" and

Ancient Ants Adventure

By Shadow Master

"very addictive". One energized fan reported the game as "Absolutely Brilliant! This has graphics that won't look out of place in a retail game, the gameplay is great and I would pay for this game!"

The game introduces so many unique features, such as a

personal favorite of mine, the formation selector and even the different modes play, those, and others it makes it very difficult to pick out any cons here. But, to be picky there are no evident instructions in game, which can make the several buttons and bars on the interface seem more bothersome than helpful. But any player would catch onto these after less than ten minutes of play.

Conclusion

Overall, I really enjoyed playing this game, because it captures a distinct uniqueness, and still manages to hold some comic relief, considering you are playing the role of an ant, in a very colorful world!



>>> Scores

Ratings

Gameplay
Graphics
Sound
Design
Storyline



Overall Score



Get it now!

yoyogames.com/games/show/34591

>>>Creation

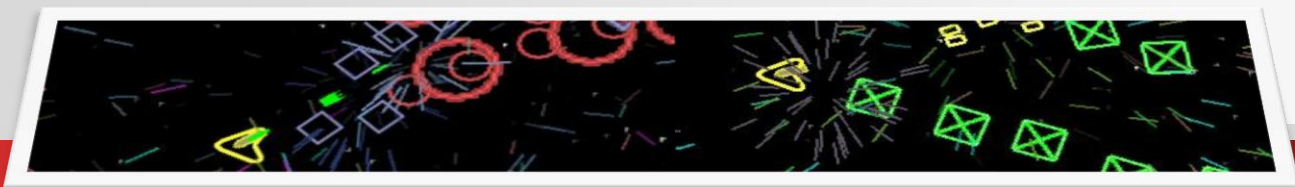
ZIO

ZIO is a space flight game created by rat5522. It is an awesome, well-designed, engaging game where you battle 14 types of enemies and an appealing graphical environment that suits the game pretty well.

It is a fun little Geometry Wars clone, but still addictive and well-made, so you should try it out.

The game is generally smooth and the known annoyances have been fixed in game updates. One disappointing thing is that ZIO uses Game Maker's built-in dialog boxes which might tick you off but shouldn't harm the addictive gameplay experience itself. Try it out.

yoyogames.com/games/show/26402



The Adventures of Cendah

The Adventures of Cendah is a Zelda-style RPG game with spells, weapons, items, and inventory created by KingDiz.

Among the 'features' of the game, we have a real time battle system, an introductory tutorial system, well-styled RPG graphics and sound, and a generally well-made story.

The Adventures of Cendah is a cool, entertaining game, well-received on YoYo Games and the Game Maker



Community. There are many quests in the game which you need to solve in order to follow the gameplay to the end. You can try this well-made game on YoYo Games.

yoyogames.com/games/show/26930

World in War

World in War is an excellent war-game by Sokota and the SkyFire team. The game features excellent combat, a variety of vehicles, excellent graphics fitting to the atmosphere, and great building and setting design. The game is still in early development stages with very basic

enemy AI and little action at times. Another problem that needs to be addressed would be the utter lack of background music as well as the need for more sound effects. Promising project overall, check it out!

gmc.yoyogames.com/?showtopic=369060



Pokémon Beta

The game caught my attention when I saw sunstroke's signature (the creator of the game) and saw "I can't believe Pokémon graphite is getting more attention than my game"! I set out to try the game out, and when I downloaded it and tried it out; I came to one conclusion: this is actually pretty good.

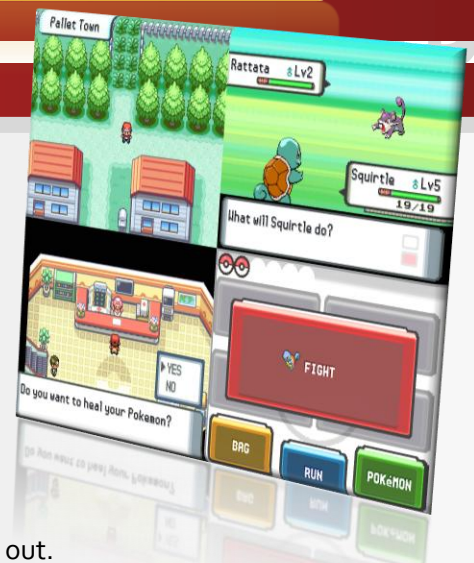
Pokémon Beta is graphically a GBA-style game though its battle engine (and other portions) features a touch section of the window, clearly a DS-style method of interactions. I

like that.

The game already features a battle system as well an overworld and all of the basic features of a typical Pokémon game.

Good job so far! Try it out.

gmc.yoyogames.com/?showtopic=378311



GOODBYE!

And that's Issue 14 of MarkUp Magazine! We hope you enjoyed reading it; we sure did enjoy making it! We took a break and were away for a couple of months but we hope you agree that we have returned with an excellent, versatile, and content-packed issue that made it worth the long wait!

In this issue, we return with the "The Making of..." articles, giving you a very comprehensive look on the game development process, we also returned to the old pace and type of technical content we used to serve that made our readers love us to begin with, and have emphasized – by popular demand – our reviews and interviews section in the magazine. We also have a much improved graphical feel overall with our designer. While Suhaib has worked on a couple of issues previously, we only just agreed on the proper feel/style to use for MarkUp Magazine in this issue, which is why you'll see the amazing cover and inner-issue (headings, etc.) graphical work that he has done for us. Some might not like the color scheme for this issue; red *is* bold after all, but the color scheme will be continuously updated as well as the graphics art in general for the upcoming issues.

MarkUp Magazine is made, maintained, and supported by people like you! So, if you like us, make sure you support us in a multitude of ways; you can [contribute](#) to MarkUp Magazine and submit content such as tutorials or editorials that you have written, or you can [apply to staff](#) and become a fulltime writer to the magazine. Please share all your opinions and feedback regarding this issue, MarkUp Magazine in general, and anything else on our [GMC topic](#) or contact us by [mail](#)! Thanks a lot for supporting MarkUp Magazine and making it what it is today!

The MarkUp Staff■

[GMking.org](#) is the parent network for **MarkUp Magazine**. It is constructed as to behave like a centralized portal that links to the four main aspects of GMking.org's projects: The GMking.org Site [which is now a sub-site of the main gmking.org page], The GMking.org forums, GMPedia.org, and MarkUp magazines. Visit the site for MarkUp's entire set of sister projects! Also check out [GMPedia.org](#), a sister project of MarkUp Magazine: a comprehensive wiki and encyclopedia for game developers.

GMking.org *Let them make games!*

MarkUp is an open publication made possible by the contributions of people like you; please visit markup.gmking.org for information on how to contribute. Thank you for your support!

©2008 MarkUp, a GMking.org project, and its contributors. This work is licensed under the Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-nd/3.0/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA. Additionally, permission to use figures, tables and brief excerpts from this work in scientific and educational works is hereby granted, provided the source is acknowledged. As well, any use of the material in this work that is determined to be "fair use" under Section 107 or that satisfies the conditions specified in Section 108 of the U.S. Copyright Law (17 USC, as revised by P.L. 94-553) does not require the author's permission.

The names, trademarks, service marks, and logos appearing in this magazine are property of their respective owners, and are not to be used in any advertising or publicity, or otherwise to indicate sponsorship of or affiliation with any product or service. While the information contained in this magazine has been compiled from sources believed to be reliable, GMking.org makes no guarantee as to, and assumes no responsibility for, the correctness, sufficiency, or completeness of such information or recommendations.